**Dissertation presented for the degree of Master of Science in Engineering**

# Gesture Recognition with Application to Human-Robot Interaction

Ra'eesah Mangera

February, 2015




University of Cape Town
Department of Electrical Engineering
Supervisors: A/Prof. F. Nicolls (UCT) and
Mr F. Senekal (CSIR)

# Declaration

I declare that this dissertation and the work presented in it are my own work.

I confirm that:

1. This work was done mainly while registered for a MSc at the University of Cape Town.
2. I have not previously, in its entirety or in part, submitted this dissertation for obtaining any qualification.
3. Where I have consulted the published works of others, this is always clearly attributed.
4. I have acknowledged all main sources of help.
5. Parts of this work has been published as:

    a) R. Mangera, "*Static Gesture Recognition using Features Extracted from Skeletal Data*", Proceedings of the Twenty-Fourth Annual Symposium of the Pattern Recognition Association of South Africa, Aukland Park, South Africa, 2013.

    b) R. Mangera, F. Senekal, and F. Nicolls, "*Cascading Neural Networks for Upper-body Gesture Recognition*", International Conference on Machine Vision and Machine Learning (MVML), Prague, Czech Republic, 2014.

_____

Signature

_____

Date

# Abstract

Gestures are a natural form of communication, often transcending language barriers. Recently, much research has been focused on achieving natural human-machine interaction using gestures. This dissertation presents the design of a gestural interface that can be used to control a robot. The system consists of two modes: far-mode and near-mode. In far-mode interaction, upper-body gestures are used to control the motion of a robot. Near-mode interaction uses static hand poses to control a graphical user interface. For upper-body gesture recognition, features are extracted from skeletal data. The extracted features consist of joint angles and relative joint positions and are extracted for each frame of the gesture sequence. A novel key-frame selection algorithm is used to align the gesture sequences temporally. A neural network and hidden Markov model are then used to classify the gestures. The framework was tested on three different datasets, the CMU Military dataset of 3 users, 15 gestures and 10 repetitions per gesture, the VisApp2013 dataset with 28 users, 8 gestures and 1 repetition/gesture and a recorded dataset of 15 users, 10 gestures and 3 repetitions per gesture. The system is shown to achieve a recognition rate of 100% across the three different datasets, using the key-frame selection and a neural network for gesture identification. Static hand-gesture recognition is achieved by first retrieving the 24-DOF hand model. The hand is segmented from the image using both depth and colour information. A novel calibration method is then used to automatically obtain the anthropometric measurements of the user's hand. The k-curvature algorithm, depth-based and parallel border-based methods are used to detect fingertips in the image. An average detection accuracy of 88% is achieved. A neural network and $k$-means classifier are then used to classify the static hand gestures. The framework was tested on a dataset of 15 users, 12 gestures and 3 repetitions per gesture. A correct classification rate of 75% is achieved using the neural network. It is shown that the proposed system is robust to changes in skin colour and user hand size.

# Acknowledgements

First and foremost, all praise is due to Allah, the most merciful, the most gracious.

I would like to thank my supervisors Dr Fred Nicolls and Fred Senekal for their guidance, advice and encouragement throughout my MSc studies. I have benefited immensely from their knowledge. Special thanks to Benjamin Rosman for helping with the editing, I really appreciated your discerning eye.

My deepest gratitude to my parents and sister for their unwavering support, motivation and encouragement.

I would also like to thank my colleagues at MIAS and everyone else who volunteered and allowed me to record them for my data set.

# Contents

# List of Figures

# List of Tables

# 1. Introduction

Scientists predict that within the next 20 to 30 years robots could be a part of our daily lives, assisting humans with various tasks [1]. Already, they can be found vacuuming homes, mowing the lawn and more recently assisting customers in stores [2]. Therefore, it is essential that the communication between robots and humans, or human-robot interaction (HRI), be as natural as possible. To this end, there has been much research focused on imbuing robots and computers with the ability to understand human gestures. A gestural interface for controlling computers was envisioned as early as 1980, with researchers at MIT developing a voice and gesture interface to move shapes about a display screen [3].

This chapter introduces the basic terminology and definitions that are used in this work (Sections 1.1 to 1.3), presents an overview of the designed system in Section 1.4, and provides an analysis of the contributions in Section 1.5. An outline of the dissertation is provided in Section 1.6.

## 1.1. Gestures

A gesture is defined as a bodily action that conveys intent, or a movement that is deliberate or communicative [4]. Gestures can be performed using any part of the body, from the arms, as in a waving gesture, the head, as in a nod, or even the face. Gestures are thought to be one of the most natural forms of communication, particularly in noisy environments or where speech is not possible. There are four main classes of gestures: deictic, iconic, metaphoric, and beat gestures [5]. *Deictic* gestures are pointing gestures. They refer to physical locations in space. *Iconic* gestures represent the features of an action or event. An example of an iconic gesture is holding your hands together to form a pistol whilst describing shooting something. Gestures that represent concepts with no physical form are *metaphoric,* such as the waving of hands to emphasise the complexity of what is being discussed. Various speakers use *beat* gestures such as banging a fist on a table top to emphasise points. These categories are not mutually exclusive, and gestures may typically fall into more than one category. Gestures used in gesture interfaces are usually iconic, metaphoric or deictic.

A further classification is made between static and dynamic or temporal gestures. Static gestures occur at a single instance in time and are also known as *poses.* They

can be captured by a single image. *Dynamic* or temporal gestures occur over a certain period of time and can be represented by a sequence of images or a video.

This work focusses on both dynamic gestures, performed using the upper body, and static hand gestures where most of the information lies in the hand configuration.

## 1.2. Gesture Recognition

Gesture recognition is one of the core components of what developers refer to as a perceptual computer interface. It is the detection and interpretation of gestures, using a computing device. A typical gestural interface consists of a sensor, comparator and actuator. The sensor measures the environmental state such as light, proximity or motion. The sensor sends the information it receives to a comparator which is usually a computer algorithm. The comparator analyses the information, extracting features that are then used to classify the gesture based on prior knowledge. Based on the class of gesture a specific action is performed by an actuator which operates in the environment. See Figure 1.1 for a depiction of the full system.

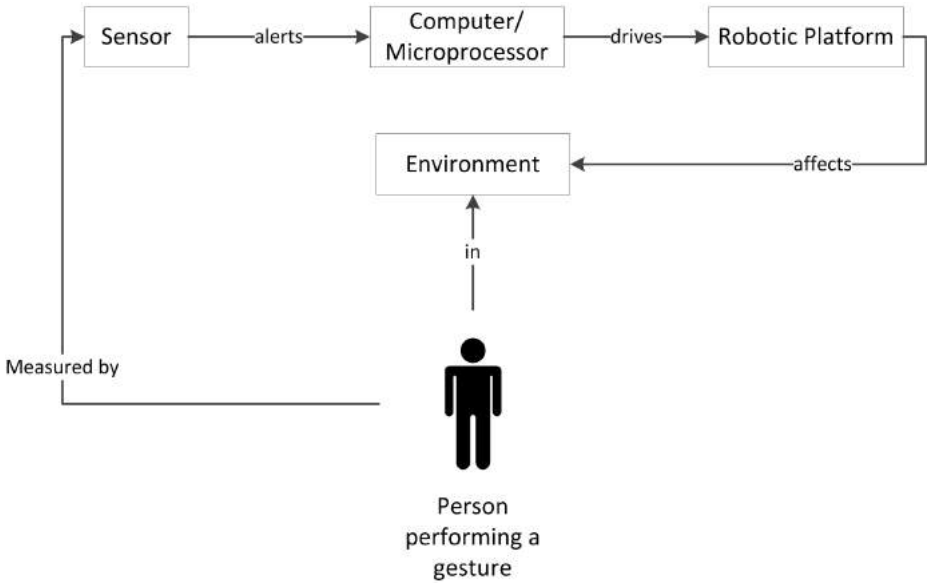



**Figure 1.1.** A typical gestural interface for communicating with a robot.

The comparator component of the system can be further split into two phases — a training phase and a prediction phase.

Typically the sensors stream data continuously. Therefore the raw data must be *preprocessed* to extract meaningful information, such as the poses or motion of the hands, from the signal. A set of features which uniquely define a gesture are then extracted. This is known as *feature extraction*. These features may be the position of body joints, the shape of the hand, or the joint angles. A set of gestures, referred to

as the *gesture lexicon*, is defined. The features of these gestures are learnt during the t*raining phase* and used to train a model which is later used for gesture classification. In the *prediction phase* the real time sensor data is also preprocessed, features are extracted and these are classified based on the trained model. The output of the classifier then dictates what action should be taken. This flow of information is depicted in Figure 1.2.

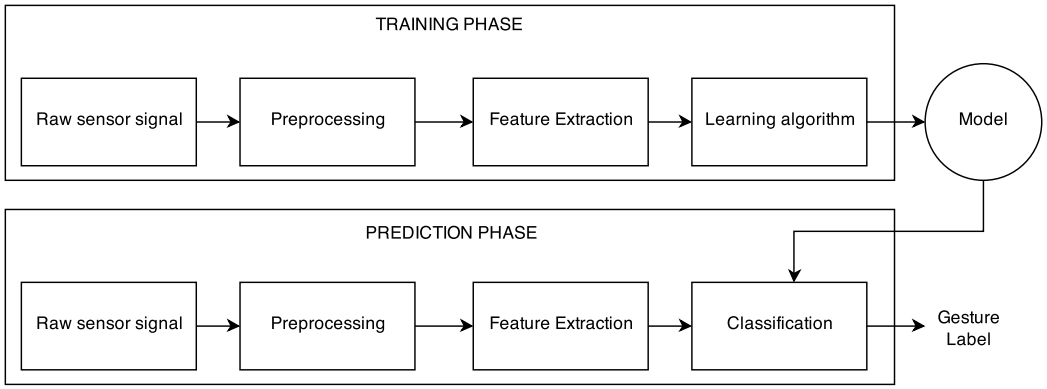



**Figure 1.2.** The phases in the gesture recognition component of a gestural interface.

According to Wachs et al. [6], the basic requirements for any gesture recognition system are:

1. Responsiveness: the system must be able to recognise gestures almost instantaneously (a maximum delay of 45 ms) as a slow system is impractical.
2. User adaptability and feedback: the majority of gesture recognition systems have a defined number of gestures which the system is able to identify. These gestures are programmed through an offline classifier algorithm. The challenge is to provide a classifier that is able to generalise a gesture from minimal training samples.
3. Learnability: the gestures used to control the system should be easy to remember and execute.
4. Accuracy: the system must be able to first detect if the hand or body is within the view, track the hand from frame to frame, and match the gesture to learnt templates (recognition).
5. Intuitiveness: gestures used in the system should be intuitive in order to mimic communication between humans. For example a closed fist with the thumb up could represent "OK". This is strongly dependent on cultural background and experience.
6. Lexicon size: a lexicon is a dictionary of the gestures used in the system. Ideally increasing the number of signs in the system should affect the performance and accuracy of the system as little as possible.

7. Garment and environment requirements: the system should not require the user to wear additional aids or to be wired to a device, and in terms of background and illumination the environment should not need to be fixed.
8. Reconfigurability: hands are different in size, shape and skin colour, thus the gesture recognition system should be invariant to these variations.
9. Mobility: many systems are dependent on the assumption that the user stands in a fixed position. For many applications this is not a valid assumption.
10. Unintended gestures: the system must be able to distinguish between intentional and unintentional gestures.

These ten requirements are integral to the development of a robust, reliable and accurate gesture recognition system.

## 1.3. Human-Robot Interaction

Human-robot interaction (HRI) is the study of communication between robots and humans. It exists at the intersection of the fields of artificial intelligence, computer vision, robotic design, social science, and the humanities. Robots are increasingly becoming involved in more complex tasks and activities, sometimes requiring interaction with people to complete these tasks. This has given rise to the field of HRI, the study of how to design and implement robotic systems that can interact with a human environment in a safe and efficient manner.

HRI is a challenging field because the system needs to be able to perceive, understand and react to human activity in real time. Challenges include:

- There is a limitation on the size of the gesture recognition system, it must be able to fit on the robot.
- As both the robot and the human are mobile, static backgrounds cannot be used for segmentation and a a fixed camera location cannot be assumed.
- The robot mobility could lead to drastic changes in environmental conditions, such as lighting.
- The system must be able to work in real-time. Ideally, there must not be a perceivable lag between the user performing a gesture and the robot response.

These challenges should be taken into consideration when designing a HRI system.

## 1.4. System Overview

An overview of the system is depicted in Figure 1.3. This diagram shows the integration of the components presented in the subsequent chapters.

There are two main components:

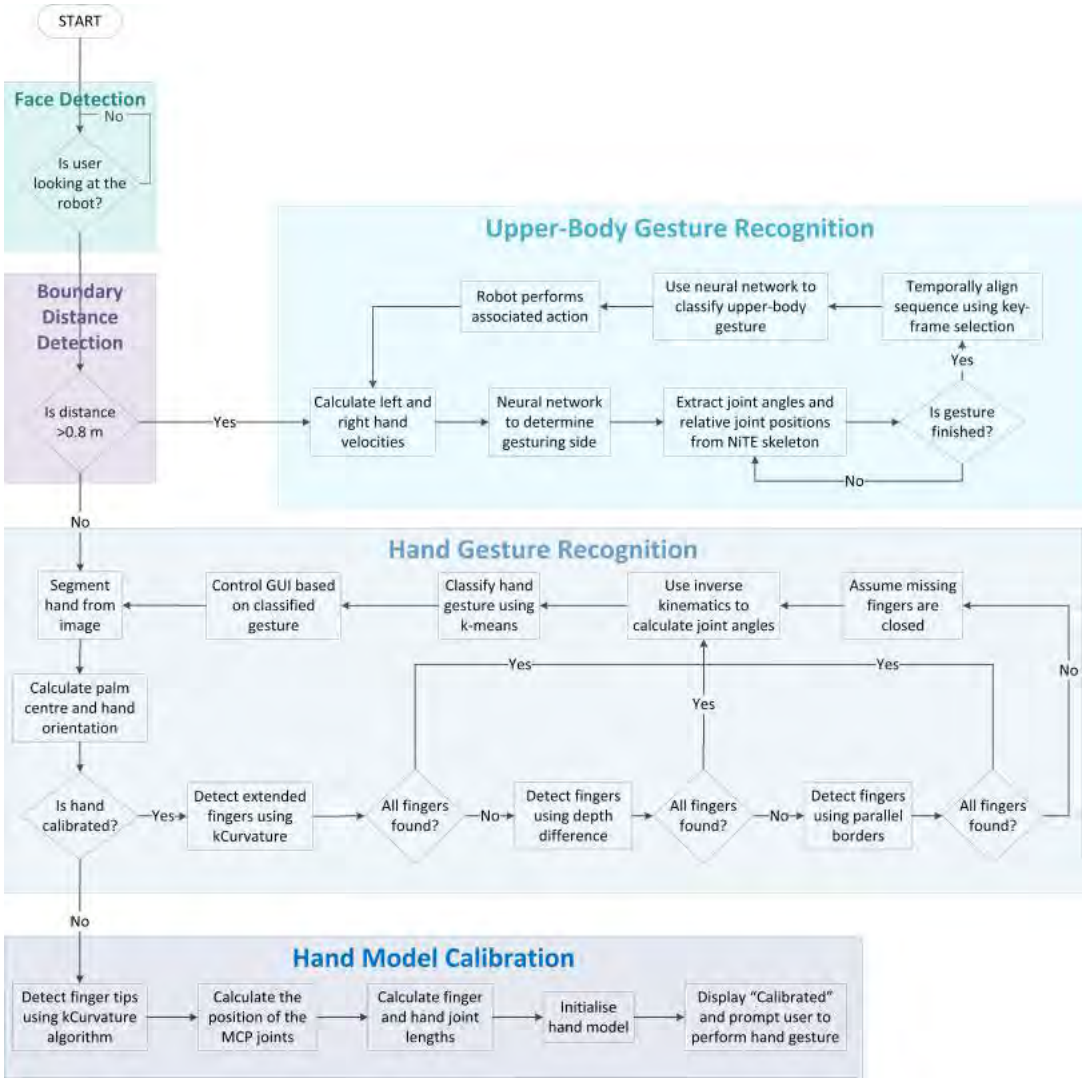



**Figure 1.3.** System overview for a human robot interaction (HRI) system.

- An upper body gesture recognition system that uses joint angles extracted from the NiTE skeleton as input to a neural network that classifies gestures performed using either the left or right arm.
- A hand gesture recognition system that uses the position of the fingertips to calculate the hand joint angles, which are subsequently used as input to a $k$-means classifier that determines the static hand pose.

First, a Viola-Jones face detector is used to determine if a user is looking at the robot. Detection of a frontal face triggers the gesture recognition system. The Viola-Jones face detector is described in Section 7.1.1.

The depth map is then used to calculate the distance between the robot and user. If this distance is greater than a threshold (described in Section 7.1.2) then the upper body gesture recognition is triggered, otherwise hand gesture recognition is

performed. The distance is chosen based on the maximum distance that a user will be able to see the GUI attached to the robot.

If the upper body gesture recognition system is triggered, the velocity of the left and right hands are calculated. These velocities are used as an input to a neural network that is then used to determine the gesturing side, as described in Sections 5.3 and 6.1.1. The relative position and joint angles of the arm are extracted from the user skeleton as they are robust to variations in scale and rotation. These features, described in Chapter 5, are then temporally aligned as described in Section 5.4. Alignment is necessary to ensure that the most important frames from the gesture sequence are input to a classifier. Two classifiers are compared for upper body gesture recognition — a neural network presented in Section 6.1.1 and a hidden Markov model presented in Section 6.1.2. These classifiers are chosen based on their ability to model temporal data such as gesture sequences. It is found that the neural network achieves a higher classification compared to the hidden Markov model. Based on the classified upper body gesture, the robot performs an action defined in Figure 3.2.

For hand gesture recognition, the hand is segmented from the image as illustrated in Figure 4.2. The hand segmentation combines depth, colour and hand orientation information improving segmentation performance. The characteristics of fingers, namely the curved fingertips, parallel borders and the fact that tips always occur as depth differences are used to detect fingertips as described in Section 4.2.2. If a fingertip cannot be detected then it is assumed that the finger is closed (fully flexed), and use the joint angles associated with a closed finger. Inverse kinematics presented in Section 4.2.1 are used to calculate the finger joint angles. Joint angles are ideal features as they are rotation and scale invariant implying that they can be used for a broad range of users. The use of hand joint angles fulfils the reconfigurability requirement for a successful gesture recognition system. The hand gesture is classified using the classifiers outlined in Section 6.2. Two classifiers are compared— a neural network and a $k$-means classifier. The performance of the neural network exceeds that of the $k$-means classifier. The hand gesture is used to control a graphical user interface with actions depicted in Figure 3.1.

In addition, there is a hand calibration component that automatically calibrates the finger joint lengths. This is needed to initialise the hand model. Hand calibration is illustrated in Figure 4.16.

## 1.5. Contribution of this Dissertation

This dissertation describes the design of a gestural interface which can be used to interact with a robot, both when it is far away (far mode) and close to the user (near mode). This interaction is achieved by capturing human behaviour using a depth sensor, as follows. First, all users in the camera's field of view are segmented

from the image stream. Frontal face detection is then employed to determine if a user is looking directly at the robot. This is used as a trigger gesture, activating the interaction module. Depending on the distance of the user from the robot, either the near mode or the far mode module is invoked.

Far-mode interaction is used to control the movements of the robot, such as turn or go away, and near-mode is used to interact with a display that is attached to the robot. In far-mode interaction, the position of the upper body joints are recorded. A gesture spotting module is used to determine whether a gesture is being performed or not as well as its time boundaries. A neural network is then used to assign a class label to the gesture. In response to this label, the robot performs a particular task.

In near-mode interaction the hand is segmented from the image. All fingertips are then identified, including those that are close together or not separated from the palm. The fingertip locations and the palm and wrist position are used to solve the inverse kinematics and estimate the finger joint angles for the hand. These angles are the features utilised by a neural network to classify the static hand gesture.

In short, the main contributions of this dissertation are:

- Proposing a complete gestural interface that can be used for both near and far interaction with a mobile robot.
- Successfully detecting dynamic upper body gestures for a variety of users without prescribing which side of the body should be used for the gestures and no additional training and calibration.
- Successfully detecting hand poses by identifying the location of all fingertips. Using inverse kinematics to determine joint angles increases the robustness of the system so a variety of users can use it. In addition, the high-level features can be used to describe a large number of hand poses.

The proposed system is evaluated using both publicly available datasets, as well as a recorded dataset.

## 1.6. Dissertation Overview

The remainder of the dissertation is organised as follows:

Chapter 2 presents a review of the existing literature in the field of gesture recognition with emphasis on gesture recognition for human-robot interaction (HRI). The various approaches employed in each phase of a gesture recognition system are discussed, highlighting the strengths and weaknesses for each method.

Chapter 3 details the acquisition of data used to evaluate the designed system. It contains a full review of existing datasets and a description of the dataset collected for both upper body gestures and hand gestures.

Chapter 4 discusses the features extracted for hand gesture recognition. This includes the strategy used for hand segmentation and the method used to extract hand joint angles from an image. Results for the hand gesture features are presented.

Chapter 5 presents the feature extracted for upper body gesture recognition in particular the extraction of joint angles and a novel key-frame selection method used to temporally align gesture sequences. The temporal alignment is quantitatively and qualitatively evaluated and the results of the evaluations are reported.

Chapter 6 discusses the theory of hidden Markov models (HMMs) and artificial neural networks (ANN) that are used for gesture classification. The application of these algorithms to gesture recognition is presented.

Chapter 7 provides an evaluation of the integrated gesture recognition system. The results for the integrated system are presented as well as a summary of the results for each component of the system.

Chapter 8 concludes the dissertation providing a description of possible future work and underlining the main contributions.

## 1.7. Summary

This chapter introduced the terminology used throughout the dissertation. An overview of the designed system was provided and the contributions of this work were highlighted. An outline of the dissertation was also provided.

# 2. Literature Review

Gesture recognition is a growing research area due to its wide range of applications, ranging from medical systems and assistive technologies, entertainment, crisis management, disaster relief, human-robot interaction and many more. Among these areas, the most common application is human computer interaction (HCI). It aims to replace the traditional keyboard and mouse interfaces with a more natural interface. A large amount of the research in gesture recognition is dedicated to sign language translation, including recognition of both static alphabets and dynamic word gestures.

This survey is organised as follows. The typical sensors used are presented in Section 2.1. Feature and classification methods employed are discussed in Section 2.2. A review of gesture recognition systems used for HRI can be found in Section 2.3. Dynamic upper body gesture systems and static hand pose systems are examined independently.

## 2.1. Sensors used for Gestural Interfaces

Two main types of sensors are used for gesture recognition, glove-based sensors and vision-based sensors.

In glove-based systems the user is required to wear a glove fitted with sensors such as accelerometers and flex sensors [7, 8, 9, 10, 11, 12, 13]. These devices directly measure the hand or arm joint angles and spatial positions. The gloves may be wired or wireless. The wired system requires the user to be in close proximity to the computer. Glove-based systems restrict natural gestures as users are required to wear additional aids that may hamper natural movements. These systems also require the glove to be calibrated for each new user, limiting the reconfigurability of the interface. However, they are typically more accurate.

Whilst there are various application specific uses for the glove-based systems, they are not a "natural" method of interacting with a robot. This shortfall has driven a large proportion of gesture recognition research toward vision-based systems. These approaches use single or multiple cameras and more recently depth sensors such as the Microsoft Kinect to capture and interpret gestures. The advantages of vision-based sensors are numerous: they are simple, low cost and do not need to be adjusted for each user. However, they suffer from the limitation that the gesture must be

performed in the camera's field of view in order to be detected. In addition, depending on the distance of the user from the camera the hand may only be represented by a few pixels, making it challenging to extract useful features for hand gesture recognition.

### 2.1.1. Vision-based Sensors

Vision-based sensors use single or multiple cameras and depth sensors such as the Kinect to capture the gesture. Vision-based sensors suffer from the limitation that the gesture must be performed in the camera's field of view in order to be detected.

#### 2.1.1.1. RGB Camera

Prior to the release of the Microsoft Kinect, vision-based gesture recognition used ordinary colour or greyscale images. These typically focused on static and dynamic hand gestures. Skin colour is used as a cue to segment the hand from the image [14, 15, 16, 17]. However, as there is no depth information available, the majority of these systems require simple, solid backgrounds [18, 19, 14, 20] or that the user wear a coloured or textured wrist band to aid in segmentation [21].

Another method of gesture recognition using colour images, presented by Drake [22] and Davis and Shah. [23], requires that the user wear a glove where each fingertip or hand joint is a different colour, making identification easier.

One of the challenges of using colour as a feature is its sensitivity to illumination changes. This makes robustness difficult to achieve, motivating the use of depth sensors together with the colour images.

#### 2.1.1.2. Depth Sensors

Since the release of the Microsoft Kinect in 2010, much of the gesture recognition work has been based on depth sensors. These sensors provide depth information in addition to the RGB image captured by a traditional camera. The depth information aids in hand segmentation [24, 25, 26, 27] and the OpenNI and NiTE SDKs provide skeleton information [28, 29, 30, 31, 32]. This has made depth sensors ideal for gesture recognition. A thorough review of gesture recognition is provided by Suarez and Murphy [33]. A sensor similar to the Kinect is produced by Asus. This sensor is known as the Asus Xtion Live Pro and has the same operating principle as the Kinect. However it does not require an external power supply.

In 2013, Intel released a depth sensor for HCI known as the Creative Senz3D camera. This sensor is intended for closer range, from 0.01 m to 1 m. It uses time-of-flight to recover depth information from the scene rather than structured light employed by

the Asus and first generation Kinect sensors. Given the novelty of the sensor there are few works that use it [34, 35].

The second generation of Kinect sensors that are shipped with the Xbox One, also now use time-of-flight for depth recovery. This sensor was made available to developers in late 2014; therefore there is not much literature available [36]. However, it reportedly has a higher depth resolution and range compared to the first generation Kinect.

Another relatively new gesture device is the Leap Motion sensor. This sensor is also for close-range HCI, and is capable of tracking all fingers with an accuracy of up to a 100th of a millimetre over a range of 1 m. The operating principle is similar to that of the Asus, projecting a distinct IR pattern and using this to recover depth. Several works that use Leap Motion for HCI are presented in the literature [37, 38, 39, 40, 41].

Whilst the earlier generation of depth sensors had several flaws, including low sensor depth resolution and reduced close-range depth recovery for applications where the user is further than 1 m, the Kinect is best for HCI. The ideal system would therefore combine two sensors, one for close interaction and another for far interaction.

## 2.2. Gesture Recognition Methodology

A Gesture recognition algorithm consists of two phases: a training phase and a prediction phase. During both phases features are extracted from the data to reduce the dimensionality of the feature space. Feature extraction is discussed in Section 2.2.1. The extracted features are then used to classify the gesture. Gesture classification methods are presented in Section 2.2.2.

### 2.2.1. Feature Extraction

Feature extraction refers to the process of extracting discriminative characteristics from an image that can uniquely identify a gesture. There are two main approaches used for feature extraction, appearance-based and model-based.

Appearance-based methods use image features to model the appearance of the hand and arm. For hand gesture recognition, typical features extracted include the shape of the hand [31, 42, 43, 44], the hand area [24] and other less intuitive features such as SURF [45], SIFT [46, 47], Haar-like features [48, 49, 50] and HOG features [51, 52, 53]. The feature space is often reduced using techniques such as PCA [15, 31]. Other higher-level features such as the location of the finger-tips [54, 55, 56] and counting the number of extended fingers [57, 58, 26] can also be used for appearance-based feature extraction. The majority of these approaches can only identify fingertips

associated with extended fingers and are unable to detect fingertips if they are not on the external contour of the hand.

Model-based approaches rely on recovering the 3D kinematic model of the hand and trying to estimate all the parameters by comparing input images with the 2D appearance of the hand model. The advantage of this approach lies in the fact that many hand gestures can be defined, however these approaches are resource intensive and typically cannot operate at speeds faster than 15 frames per a second [59]. They may also require special hardware such as a GPU (graphical processing unit) [59, 60]. The most common approach to model-based hand pose estimation is to use synthetically generated hand poses to train a classifier and has been adopted by several researchers [61, 62, 63, 64].

### 2.2.2. Gesture Classification

Pattern recognition is a technique for learning and recognising patterns that exist in a dataset. They are flexible as they can often learn in a changing environment. This is not possible for rule-based systems that can only operate in the situation for which they have been designed. For this reason the majority of gesture recognition systems employ pattern recognition classifiers.

During the training phase, a machine learning algorithm uses training data to infer the structure of the data to enable classification to take place. There are many different techniques that can be used for pattern recognition. These include artificial neural networks (ANNs) [46, 65], histogram based features [19], fuzzy clustering algorithm [10], hidden Markov models (HMMs) [17, 44], adaboost [46], support vector machines [66, 43], decision trees [67], and dynamic time warping (DTW) [28]. The classification technique selected depends on the amount of training data available, the training and classification time and the ability of the learnt model to generalise any sample. For example, neural networks and HMMs are only suitable when there is a large amount of training data whereas DTW can create a model from just a few samples. In contrast, HMMs can correctly classify gestures even when there is a large intra-class variance. This is not possible with DTW as several templates are then needed.

## 2.3. Gesture Recognition for Human-Robot Interaction

The majority of existing systems only allow for robot interaction at either close distances, where the hand shape is used as an input, or further away, where features are extracted from the position of the human body, particularly the upper body. These are termed hand gesture recognition (HGR) and body gesture recognition

(BGR) respectively. The subsequent sections provide a review of these two gestural interfaces. In the field of human robot interactions (HRI) many systems have been developed for robotic control using gestures [68, 69, 70, 71, 72, 73, 74, 75, 65, 46, 76, 77, 78, 79].

What follows is a review of the most recent approaches to using gestures for HRI.

### 2.3.1. Hand Gesture Recognition

Hand gesture recognition is commonly used to control a computer interface in the near vicinity of the user. The hand is found in the input image and features are extracted, most often based on shape. Below is a review of some recent approaches used for hand gesture recognition.

A video camera is used to capture RGB images by Hasanuzzaman and Ueno [74]. The hand is extracted from the image using skin colour as a feature. Eight static gestures are classified using principal component analysis.

Yin and Xie [68] use skin colour and a restricted coulomb energy (RCE) neural network to segment the hand from an image. They count the number of fingers that are extended and the distance from the arm to train another RCE neural network to classify static hand gestures. For a gesture lexicon of size eight, an accuracy of 95% is achieved.

Thakur [72] define a lexicon of five gestures, the numerals 1 through 5. The system captures images in real-time using a colour camera. The Y-Cb-Cr colour space is used to isolate the hand from a uniform background using Otsu segmentation [80]. Peak and valley points are identified in the image using the k-curvature algorithm [81]. The numbers of peaks and valleys are used for gesture classification. An accuracy of 95.2% is achieved. This system is limited to a distinct number of gestures (six) and cannot be extended to include more as only the number of fingers is used as a feature.

Trigueiros et al. [73] use depth information to segment the hands from the image. They assume that the hands are the closest object to the camera. The orientation of the hand is then determined using a vector from the mid-point of the hand to the furthest point on the hand contour. The hand orientation is used to control the orientation of a robot whilst it is moving, and the distance between the hand centroid and image determine the robot's linear velocity.

Van den Bergh et al. [75] use the shape of the hand to control a robot. Rather than extracting high-level features, all intensity values in the depth and colour image within a bounding box around the hand are used. Average neighbourhood margin maximisation (ANMM) is used to reduce the dimensionality of the feature space. A nearest neighbour classifier is used to classify four gestures.

Wang and Wang. [46] extract scale invariant feature transform (SIFT) features from a greyscale hand image. Adaboost is used to classify a total of three gestures—

palm, fist and six. They achieve an accuracy of approximately 96%. The primary advantage of the system is robustness to viewpoint changes that may occur. Recognition accuracy is still high even when the hand is rotated up to 40 degrees. However, the lexicon size is small, and the hands are captured against a uniform background.

Xu et al. [76] present an online dynamic hand gesture recognition system. After segmenting the human body from a depth image, they use the chamfer distance to match a hand template to edges in the image. A skin colour model is then used to refine the hand region. The hand centroid is tracked through several image frames and the orientation and hand location are used for the final feature vector. Hidden Markov models are trained to distinguish between gestures. Two different datasets are used to evaluate system performance, one with six gestures and another with ten gestures. An average accuracy of 98.1% on dataset 1 and 96.1% on dataset 2 is obtained. The gestures are used to control the motion of a SIATRob robot.

Yang et al. [78] use a geometric feature to classify a total of six hand gestures. Skin colour is used to detect the hand in the colour input images. The distance between the hand centre and all points on the hand contour are then extracted. The number of peaks and valleys in this signal are used to classify a gesture. High accuracies of between 95% to 100% are achieved for three different datasets. However, like Thakur [72] the gesture set is limited to six as more gestures cannot be characterised using the defined feature vector as only the number of fingers are counted.

### 2.3.2. Body Gesture Recognition

Body gesture recognition is typically used to control the motion of a robot. Gestures are used to determine whether the robot should turn left or right, come closer or pick an object up.

Waldherr et al. [65] define a gesture lexicon of four whole body gestures. They also use skin colour to track the person but include shirt colour in their tracking algorithm. An ANN is used to train and classify the gestures. Their system achieved a gesture recognition rate of 97%. This system is not scale invariant and requires the person to be a specified distance from the robot.

Yeasin and Chaudhuri [82] describe a gesture recognition system which uses a video camera to capture temporal gestures and use these to control a robot. The system then finds the temporal signature of the gesture by dividing the gesture into subsets such that for each subset motion is only performed in one direction. The order of the directions determines the gesture performed. Finite state machines are used to classify the gesture. The system was found to perform well on synthetically generated data, but yields poor results for real images as the motion sequence is not as uniform.

Zhao et al. [69] use gestures to call an elderly service robot. Rather than using the

traditional skeleton-based approach for feature extraction they include an octree-based method that operates when skeleton generation fails. The head and hand are segmented from the image, and various geometric properties are extracted. They validate the approach for a number of different situations where the skeleton generation would fail, for example if the user is sitting on a sofa. By fusing the Kinect skeleton and octree approaches, an accuracy of over 95% is achieved for four calling gestures.

Kopanivčáková and Virčíková [83] develop a system to record dynamic gestures and discuss how these can be applied to human-centred robotics. The system captures the Kinect skeleton joints, and DTW is used to recognise gestures. However, the system is not user-independent and training data must be added for new users.

Yang et al. [77] estimate the 3D pose of the human body in a stereo camera video sequence. The relative angular position of each joint with respect to the mid-back is used as a feature vector in each frame. These are concatenated over the video sequence to form the full feature vector. A HMM is used for gesture classification. The overall system recognition accuracy is 94.9% for 14 gestures.

### 2.3.3. Discussion

In summary, few gesture recognition systems are robust enough to meet all the requirements for a real-life HRI system. The most common shortcoming is the inability to adapt to any user and the small lexicon size due to the types of features extracted. This shortcoming is especially true for HGR systems.

Whilst an attempt is made at incorporating domain knowledge into hand gesture recognition, the inability of most systems to identify fingertips even when they are not extended reduces the lexicon size. Tracking methods are successful at reconstructing the full 27-DOF hand model. However, they are computationally expensive, requiring graphical processing units (GPUs), extensive training sets or are unable to run in real time.

# 3. Data Acquisition

This chapter provides a brief overview of the publicly available gesture recognition datasets, namely hand pose and body gesture datasets, in Section 3.1. In addition, the recording of the hand pose and dynamic gesture datasets which is for evaluating the designed system in Section 3.2 is described.

## 3.1. Publicly Available Datasets

There are datasets that have been made available for download allowing authors to compare results with a common benchmark. Section 3.1.1 presents the dynamic gesture sets available and Section 3.1.2 presents the static hand pose datasets available.

### 3.1.1. Dynamic Gesture Sets

There are a number of publicly available datasets for gesture recognition, captured using a variety of sensors. These are summarised in Table 3.1. The MSRC-Kinect

**Table 3.1.** Publicly available dynamic gesture datasets

| | Source | Dataset | Abbreviation |
|---|---|---|---|
| 1 | Fothergill et al. [84] | Microsoft Research Cambridge-12 Kinect Gesture | MSRC-Kinect |
| 2 | Lin et al. [85] | Keck Gesture dataset | Keck |
| 3 | Guyon et al. [86] | ChaLearn dataset 2012 | ChaLearn2012 |
| 4 | Escalera et al. [87] | ChaLearn dataset 2014 | ChaLearn2014 |
| 5 | Celebi et al. [28] | VisApp2013 Gesture dataset | VisApp2013 |
| 6 | Bernstein et al. [30] | Cornell Military Gesture Dataset | CMU |

dataset [84] consists of 12 full-body gestures that relate to gaming, music, and dance. The main purpose of the work was in investigating methods for instructing users for gesture performance. The dataset consists of 30 subjects recorded using a Microsoft Kinect. There are total of 6244 gesture instances, approximately 500 per class. The

dataset provides only the skeleton positions of 20 joints in the human body at a sample rate of 30 Hz with approximately 2 cm accuracy in joint positions. The environmental conditions are not specified. The majority of participants were right handed and male. As this dataset is not restricted to only upper body gestures, it is not used for performance evaluation of the proposed system.

The Keck dataset [85] consists of 14 military signals performed using only the upper body. The dataset was collected using a standard colour VGA camera with a resolution of $640 \times 480$. The environmental conditions for the training set and test set were different. For the training set the camera remained in a fixed position, and the background is static and uniform. In the test dataset the camera was moving, and the background contains clutter and other moving objects. There are 21 instances of each gesture: nine training and twelve testing. Each gesture was performed seven times by three different participants. This dataset has no depth information available; therefore, it could not be used for testing the proposed system.

The main purpose of the ChaLearn2012 dataset [86] was a competition for one-shot training: the problem of learning from only one training sample. The gesture lexicon is vast, with 86 different gestures covering a wide range of application areas including Italian sign language, diving signals and many more. The gestures are captured using a Kinect camera with a resolution of $240 \times 320$, at a low frame rate of just ten frames per second. There was a total of 20 participants, recorded in front of a fixed camera. No information is provided with regard to the background and lighting conditions. Both the RGB and depth data were provided to challenge participants. However, joint positions were not recorded and therefore this dataset could not be used for testing.

The ChaLearn2014 dataset [87] focused on recognising gestures from several instances performed by different users. The gesture lexicon consists of 20 Italian cultural signs performed using the upper body. The data is collected using a Kinect sensor and includes the depth map, RGB image and skeleton information for 7754 gesture instances. There are approximately 388 samples per gesture.

The VisApp2013 dataset [28] consists of eight dynamic gestures performed using the upper body. There are four unique gestures, performed using the left and right sides of the body. The skeleton from the Kinect SDK is recorded. The SDK tracks 15 joints in the human body in real time at a frame rate of 30 frames per second (fps). Unlike the majority of datasets the data was recorded using users who were unfamiliar with the system, resulting in a dataset which is noisy in terms of gesture starts, gesture ends, and gesture duration. There are only 28 samples of each gesture, eight samples for training and twenty for testing. This dataset is used to evaluate the performance of the body gesture recognition system.

The CMU dataset [30] consists of fifteen dynamic gestures, namely, “action”, “advance”, “attention”, “charge”, “cover”, “crouch”, “rally”, “shift fire”, “point of entry”, “confused”, “hurry”, “sneak”, “out of action” and “come”. The gestures are performed using the right arm and the skeleton from the NiTE SDK is recorded.

Three users perform each of the 15 gestures ten times giving a total of 450 gesture instances. The dataset is recorded using a Kinect sensor and the joint positions of the body are saved. This dataset is used to evaluate the performance of the body gesture recognition system.

### 3.1.2. Hand Gesture Datasets

The majority of publicly available hand gesture sets are recorded using only an ordinary camera. Thus, there is no depth information available. The datasets that have both colour and depth information are summarised in Table 3.2.

**Table 3.2.** Publicly available hand pose datasets.

| | Source | Dataset | Abbreviation |
|---|---|---|---|
| 1 | Ren at al. [42] | NTU Microsoft Kinect Hand Gesture dataset | NTU |
| 2 | Tompson et al. [88] | NYU Hand Pose dataset | NYU |
| 3 | Qian et al. [70] | Microsoft Research Asia Hand Pose dataset | MSRA |

The NTU dataset [42] contains both colour and corresponding depth from the Kinect sensor. It contains ten gestures from ten subjects with ten repetitions per gesture. Therefore there are one thousand gesture instances. However, no skeleton tracking information is provided and the dataset cannot be used for evaluating the hand gesture algorithms.

The NYU dataset [88] consists of colour and depth images from three different views, one frontal and two side. There are only two users recorded, and rather than defining specific gestures the dataset contains a wide range of hand poses. As there is no ground truth, this dataset is hard to use for effective evaluation.

The MSRA dataset [70] is recorded using an Intel Creative depth camera. This dataset consists of six subjects performing various rapid gestures, with ground truth data for 2400 frames. Rather than a gesture label, the ground truth is the position of the five fingertips and the wrist. However, as there are no gesture labels this dataset cannot be used to evaluate the designed gesture recognition system.

## 3.2. Recorded Dataset

To evaluate the performance of the gesture recognition system a dataset is recorded. This dataset is particularly important for the hand gestures as none of the publicly available datasets are suitable. Ethics clearance obtained from the CSIR and UCT can be found in Appendix B.

### 3.2.1. Gesture Lexicon

Two types of gestures will be used for robotic control — far-gestures and near-gestures. Far-gestures are used to control the motion of the robot whilst Near-gestures are used to control the computer interface attached to the robot.

The hand poses are seen in Figure 3.1. The twelve near-gesture actions defined are:

- *Calibrate*: obtain user finger lengths and calibrate the hand model.
- *Mouse*: control the position of the mouse on the screen using the user's hand.
- *Select*: select the item where the mouse pointer is located.
- *Pan Left*: move the screen display to the left.
- *Pan Right*: move the screen display to the right.
- *Scroll Up*: move the screen display up.
- *Scroll Down*: move the screen display down.
- *Numbers 1–5*: used to enter numbers.

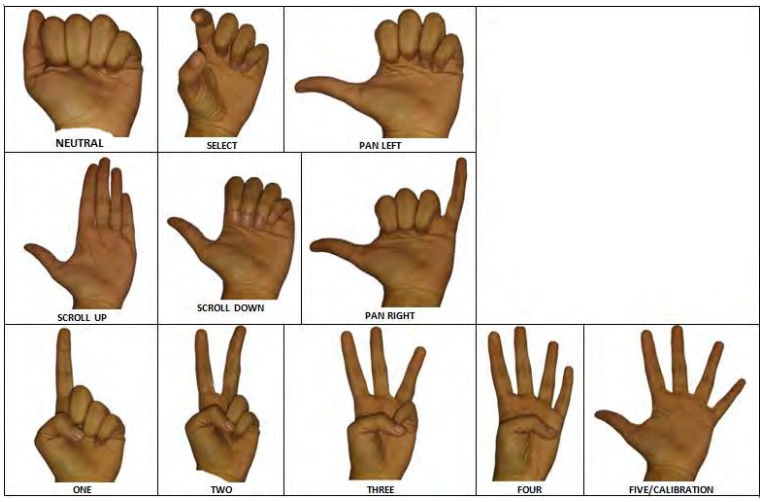



**Figure 3.1.** Static hand gestures for close HCI in the recorded dataset.

The following ten far-gesture actions are defined. The corresponding gestures can be seen in Figure 3.2:

- *Start/ Wake Up*: this will wake up the robot. Once this gesture has been performed the robot will await further instructions.
- *Stop*: stop the robot motion. This gesture should function as a dead man switch/emergency stop in that it overrides all previous commands.
- *Come Here*: indicate to the robot to come towards the user. The robot will stop at a specified distance from the user.

- *Go Away*: instruct the robot to move away from the user.
- *Move Left*: indicate the robot should move left from its current position.
- *Move Right*: specify that the robot must move right from its current position.
- *Follow Me*: indicate that the robot should follow the gesturing subject.
- *Take a Picture*: command the robot to use the on-board camera to take a picture of the current scene.
- *Wander*: indicate that the robot should wander around the room.
- *Teleop*: activate a joystick that can be used to control a robot.


**Figure 3.2.** Dynamic far-gestures in the recorded dataset.

### 3.2.2. Recording

An Asus Xtion Pro Live sensor, which is similar to the Kinect, was used to record the dataset. This sensor does not require external power and has a better near-field range than the Kinect. By emitting a distinct IR pattern and capturing the distortion of its image the sensor can return 3D coordinate information. Note that the distance to the camera plane is returned, rather than the actual distance to the object from the sensor. The specifications of the sensor can be seen in Table 3.3.

**Table 3.3.** Specifications of the Asus Xtion Pro Live Sensor

| Characteristic | Sensor Specification |
|---|---|
| Camera Resolution | 1280 ×1024 |
| Frame Rate | 30 Hz |
| IR Depth Resolution | 640 × 480 |
| Field of View | 70° (Diagonal), 50° (Horizontal), 45°(Vertical) |
| Range | 0.8 – 3.5 m |
| Power | Single USB 2.0 |

A total of 15 subjects were recorded in a laboratory environment. For the dynamic far-gestures, participants stood approximately 1.5 m from the sensor and for the static close hand gestures users were seated approximately 0.7 m from the sensor. Subjects were instructed using pre-recorded videos of the gesture that they were asked to replicate.

# 4. Hand Gesture Feature Extraction

Hand gestures have a broad range of application from sign-language recognition to controlling computer interfaces. Recognising hand gestures is challenging due to the high dimensionality of the problem implying that there is a large number of possible hand configurations. These difficulties make it very hard, if not impossible, to write software by hand to do this, and so pattern recognition approach is adopted. This approach involves solving several problems such as:

- automatic hand detection and segmentation, and
- extracting features that have a large inter-class variation but a small intra-class distribution (i.e. can effectively distinguish between different classes of gestures), and can be used to describe a large subset of gestures.

In this chapter, solutions to these problems are proposed. First, the use of depth and colour information for hand segmentation are discussed in Section 4.1. The extraction of high-level features, namely the joint angles for the hand, are presented in Section 4.2. An overview of hand gesture feature extraction is shown in Figure 4.1. The results for hand gesture feature extraction are also presented.

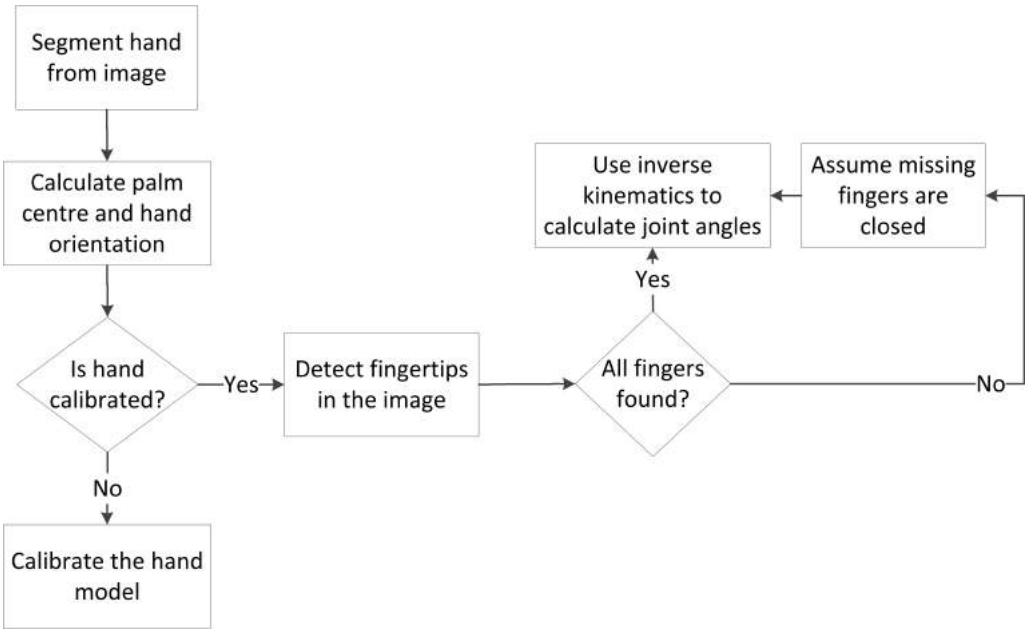



**Figure 4.1.** Overview of hand gesture feature extraction.

## 4.1. Hand Segmentation

The first step is detecting and isolating the hand from the rest of the image. This is a challenging task as the appearance of the hand is not consistent: as people have different skin colours, lighting changes in the environment, and the hand has a high number of degrees of freedom (DOF). Therefore, hand segmentation is often accomplished using skin colour models to account for variations in skin colour and lighting, and depth thresholds to account for the high DOF. The approach taken for hand segmentation is shown in Figure 4.2. The initial position of the hand is obtained from the NiTE skeleton, available from the NiTE SDK [89]. This is used to get the skin colour and hand depth of the user. The colour of all pixels in the image is then compared to the user skin colour to obtain a colour mask. Depth thresholding is employed to get a depth mask. If a pixel is within the correct depth range and matches the user skin colour then it is a hand pixel.

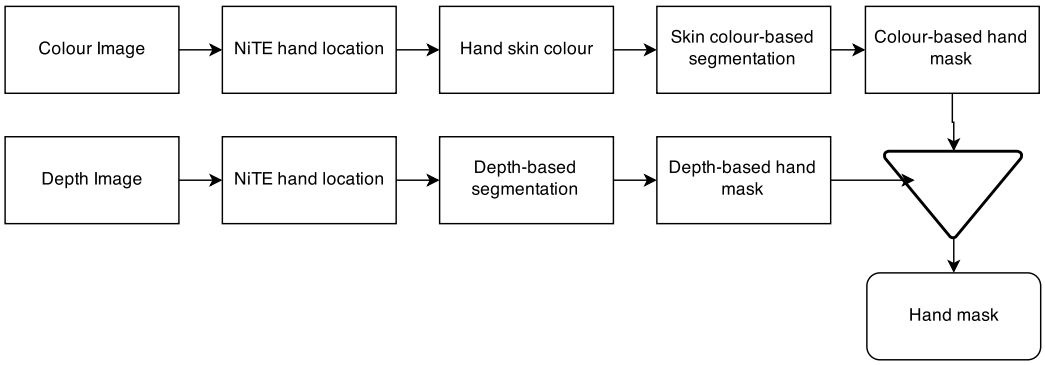



**Figure 4.2.** Hand segmentation approach.

### 4.1.1. Skin Detection

Skin segmentation is the process of classifying a pixel as being skin coloured or not. However, the appearance of skin is significantly affected by changes in illumination, and different cameras can produce different colours even for the same individual. Additionally, there is a large variation in the skin tones of people from around the world. Any skin detection algorithm must be robust to these perturbations.

One of the challenges in segmenting the hand from an image is its size compared to the rest of the image. Even at distances as close as half a metre, the total area of the hands is only approximately 10% of the image. Therefore, the hand position from the NiTE SDK is used to extract a region of interest where the hand is expected to be. The size of the ROI is proportional to the distance of the user from the camera, the closer the user, the larger the ROI. All subsequent processing is then applied only on the ROI, decreasing the processing time.

Researchers often use colour to segment human skin in pictures [26, 90, 44, 14, 32] . The two primary approaches to colour-based skin segmentation are a model-based approach and a deterministic approach. In a model-based approach, skin colour is modelled using pattern recognition techniques, such as Gaussian mixture models and naive Bayesian methods, and for each pixel the probability of it being skin is calculated. The deterministic approach uses explicitly-defined rules to determine whether a pixel is skin or not.

Both of the skin segmentation approaches can operate over a broad range of colour spaces. A full review of the various colour spaces that are commonly used for skin detection are provided in the work presented by de Campos [91].

As the system is required to work for a variety of users of different skin colours, an approach that adapts to each user is proposed. This approach relies on the initial hand position from the NiTE SDK. The colour of this pixel (the initial position of the hand) is compared to the intensity of other pixels in the image to classify if a pixel is skin coloured or not. The LAB colour space is used for colour comparisons as it is perceptually uniform, and a standard metric can be used to measure the similarity between two colours.

Like the majority of colour spaces, the CIE Lab colour space has three channels. The $L$ channel for luminance, indicating how light or dark a pixel is, and two colour channels $a$ and $b$. Unlike other colour spaces, perceived colour differences correspond to distances [92]. This implies that the similarity between two colours can be calculated in a way that resembles human colour similarity judgement. Therefore, all pixels in the ROI that are similar to the colour of the initial hand point can be found.

Let the colour of the initial hand position be $(L_i, a_i, b_i)$. To then determine whether a pixel at position $j$ with colour $(L_j, a_j, b_j)$ is classified as skin, the Euclidean distance is calculated and if it is less than a threshold $\tau_{Lab}$,as shown in Eq. (4.1),

$$\sqrt{(L_i - L_j)^2 + (a_i - a_j)^2 + (b_i - b_j)^2} < \tau_{Lab}, \tag{4.1}$$

it is classified as skin. Another method for hand segmentation is depth thresholding. The initial position from the NiTE SDK can also be used to threshold the image based on the distance of the hand from the camera. Let the depth of the initial hand position be $d_i$. For any pixel with depth $d_j$ it then belongs to the hand if the absolute difference between the distances is less than a threshold $\tau_d$, or

$$|d_i - d_j| < \tau_d. \tag{4.2}$$

Morphological operators, such as dilation and erosion, are used to fill in holes in the hand contour. These may occur at dark regions in the hand where there are shadows present.

### 4.1.2. Forearm Separation

If the user is wearing short sleeves then the forearm is also segmented as part of the hand. To segment only the hand an approach similar to the one presented by Wang and Xu [93] is adopted. This method relies on the following assumptions:

1. The wrist is located on a line tangent to the maximum inscribed circle.
2. The forearm is perpendicular to the wrist line.
3. The wrist point is located on the maximum inscribed circle at the point where the tangent is perpendicular to the forearm orientation.
4. The diameter of the palm is greater than the diameter of the wrist.

These assumptions are depicted in Figure 4.3. The grey area is the maximum inscribed circle. The green point indicates the centre of this circle which corresponds to the centre of the palm. The wrist point, indicated in red, is located at a tangent to the inscribed circle (blue line) perpendicular to the orientation of the forearm.

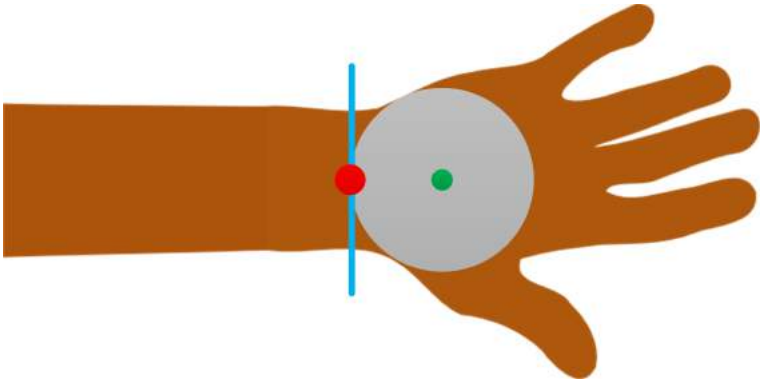


**Figure 4.3.** Segmentation of the forearm from the hand.

As noted above, the palm centre is the centre of the maximum inscribed circle. This is the circle with maximum area that fits exactly into a given set of points or contour. The maximum inscribed circle can be found using the distance transform [94]. First determine the distance to the closest pixel with intensity zero for each pixel in the image using the distance transform. The centre point is then given by the point in the hand that maximises the distance to the closest hand boundary.

The orientation is calculated as in Liang et al. [95] where it is assumed that the direction of the forearm corresponds to the eigenvector with the largest eigenvalue of the covariance matrix of the hand contour.

Consider a binary image that contains the hand mask found as described in Section 4.1.1. Contours are retrieved from the binary image using border following [96]. The covariance matrix of the points on the contour with the largest area is calculated. The covariance matrix is a measure of the correlation between $x$ and $y$ and

for 2D data, and is calculated as

$$\Sigma = \begin{bmatrix} \sigma(x,x) & \sigma(x,y) \\ \sigma(x,y) & \sigma(y,y) \end{bmatrix}. \tag{4.3}$$

The covariance matrix can also be represented as a function of its eigenvectors and eigenvalues

$$\Sigma = VLV^{-1}, \tag{4.4}$$

where $V$ is a matrix whose columns are the eigenvectors of $\Sigma$ and $L$ is a diagonal matrix whose non-zero elements are the corresponding eigenvalues. This form of the covariance matrix is obtained using singular value decomposition (SVD). The eigenvectors represent the directions of the variance of data, and the eigenvalues represent the magnitude of the variance in those directions. Therefore, the eigenvector corresponding to the largest eigenvalue represents the direction of greatest variation in the data. This eigenvector corresponds to the orientation of the forearm, as it is assumed that the area of pixels associated with the forearm will be much more than those belonging to the hand.

### 4.1.3. Results

To evaluate the performance of the segmentation scheme, a dataset of 10 images was used. These can be seen in Figure 4.4.

The dataset consists of users with different skin colours and hand sizes to test the robustness of the segmentation algorithm. RGB images, depth images and the NiTE hand position were captured using an Asus Xtion Pro Live sensor. Ground truth, containing only the hands in the image was manually marked to aid in quantitative assessment of the segmentation performance. The global-consistency error (GCE) and local-consistency error are used to evaluate the segmentation performance [97]. For a given pixel $p_i$ consider the hand segments $S_1$ and $S_2$ in the two images $I_1$ and $I_2$ that contain that pixel. If one segment is a subset of the other, then the pixel lies in an area of refinement and the error should be zero. However, if there is no subset relationship then the error should be nonzero. The local refinement error is defined in the work by Martin et al. [97] as

$$E(S_1, S_2, p_i) = \frac{|U(S_1, p_i) - U(S_2, p_i)|}{|U(S_1, p_i)|}, \tag{4.5}$$

where $U$ denotes the set of pixels in $S_1$ that contain pixel $p_i$, and $|x|$ is the cardinality of set $x$. Considering only two segments, background and skin, and the user-defined

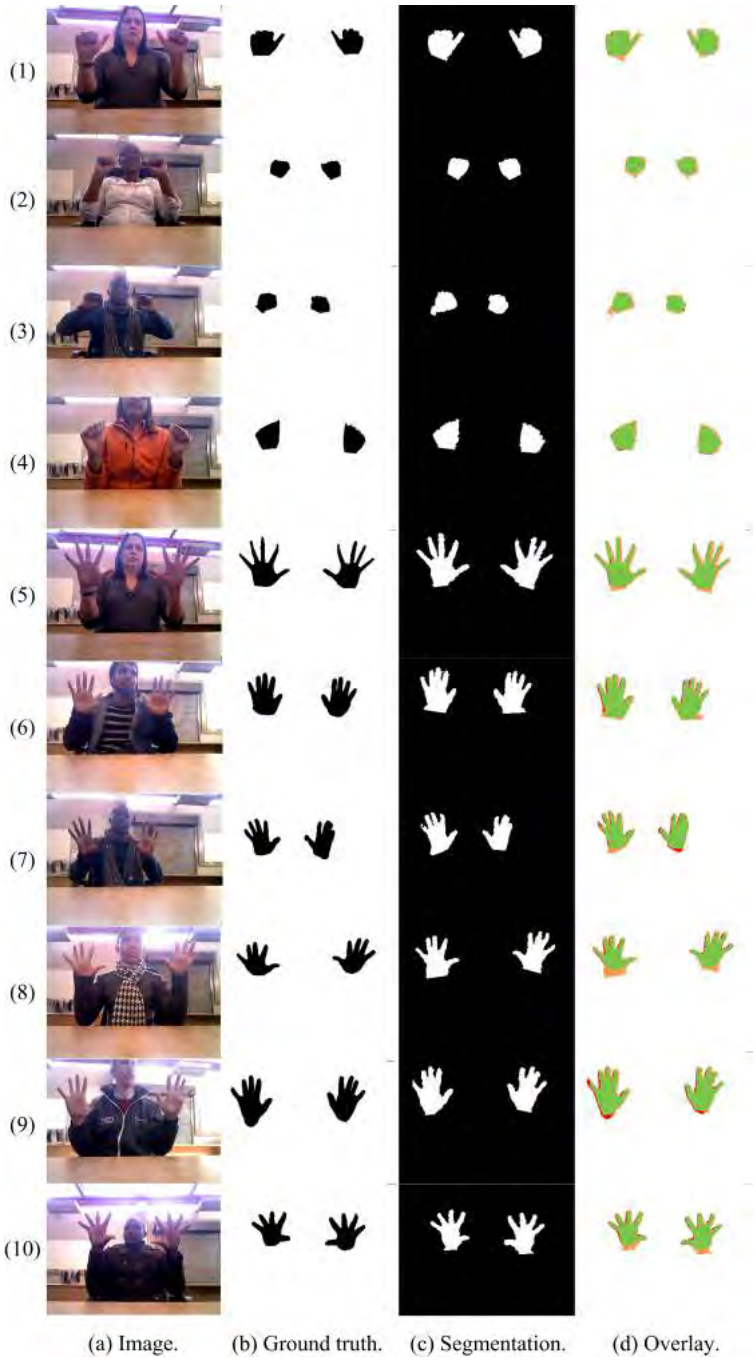



**Figure 4.4.** Dataset used to evaluate segmentation performance. (a) Original colour image. (b) Manually segmented ground truth image. Black indicates pixels belonging to the hand. (c) Results of the segmentation algorithm. White indicates pixels belonging to the hand. (d) Ground truth and segmented images overlayed. Green is correct classifications, orange is background pixels classified as skin ($S_F$), and red is skin pixels classified as background ($B_F$).

ground truth Eq. (4.5) can be simplified to the following four equations:

$$E_1 = \frac{S_F}{N_B} \tag{4.6}$$

$$E_2 = \frac{B_F}{B_T + B_F} \tag{4.7}$$

$$E_3 = \frac{B_F}{N_S} \tag{4.8}$$

$$E_4 = \frac{S_F}{S_F + S_T}. \tag{4.9}$$

The number of background pixels classified as skin is $S_F$, $S_T$ is the number of skin pixels classified as skin, $B_F$ is the number of skin pixels classified as background, $B_T$ is the number of true background pixels and $N_B$ and $N_S$ are the number of background and skin pixels in the ground truth image. The local and global consistency errors are as defined in the work by Martin et al. [97]:

$$GCE(S_1,S_2) = \frac{1}{n} \min \left\{ \sum_i E(S_1, S_2, p_i), \sum_i E(S_2, S_1, p_i) \right\} \tag{4.10}$$

$$LCE(S_1, S_2) = \frac{1}{n} \sum_i \min \{E(S_1, S_2, p_i), E(S_1, S_2, p_i)\} \tag{4.11}$$

where $n$ is the total number of pixels.

The error measures for the 10 images in the dataset can be seen in Figure 4.5. The amount of overlap between segmentation's can be seen in Figure 4.4.

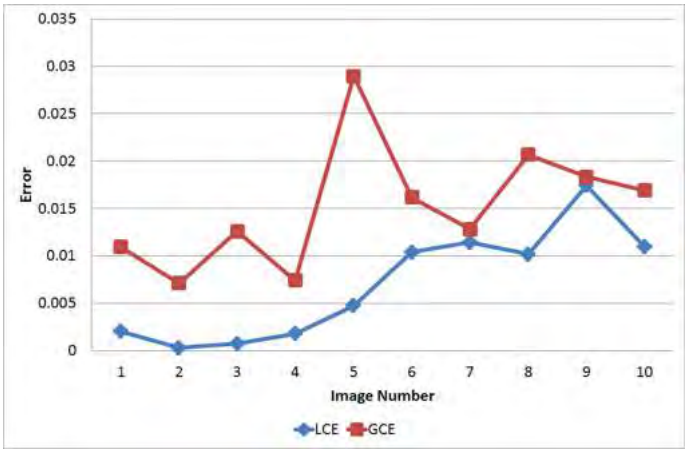
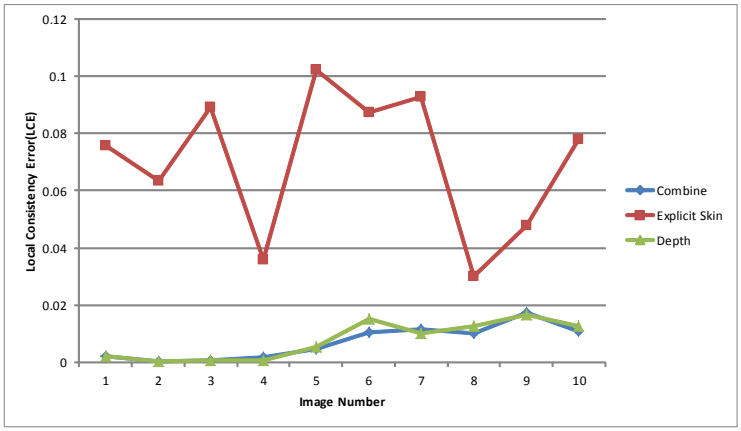


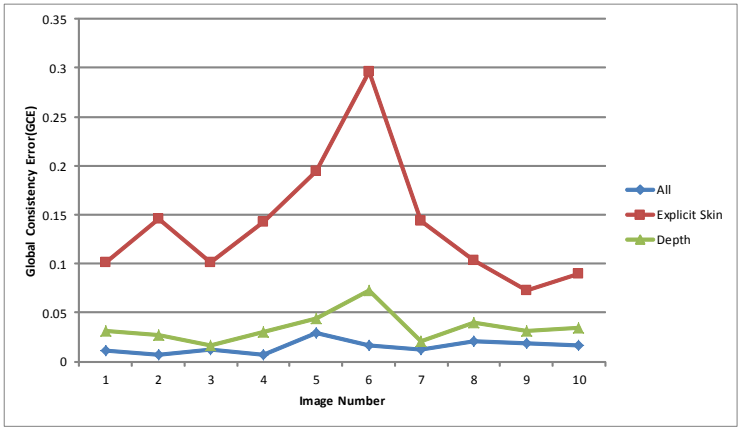

**Figure 4.5.** LCE and GCE for different images in the dataset.

As expected LCE $\leq$ GCE as GCE is a tougher measure of segmentation performance as all local refinements must be in the same direction. Details of these measures can be found in the work by Martin et al. [97]. Note that all the segmentation results have a GCE less than 3% and LCE less than 2%. False positives, or background

pixels classified as belonging to the hand, contribute the majority of the error due to the larger appearance of the hand in the depth image. In Figure 4.4, for persons (6) and (8) the wrist is larger than the palm, resulting in a larger inscribed circle. This error depends on the user's clothes. A comparison to an explicit skin colour segmentation [90] or segmentation based on depth-based methods is shown in Figure 4.6. The poor performance of the explicit skin colour-based method is due to skin-coloured objects in the environment. The combined method and the depth-based methods perform equally well in terms of both the LCE and GCE performance. In addition, the rule is unable to classify all types of skin colours, as illustrated in Figure 4.7. Skin coloured objects in the background are segmented (shown in orange) and the rule is unable to cope with a wide range of skin colours (shown in red). Depth, colour and forearm separation yield the best results.




**(a)** LCE performance.




**(b)** GCE performance.

**Figure 4.6.** Comparison of three different segmentation methods.

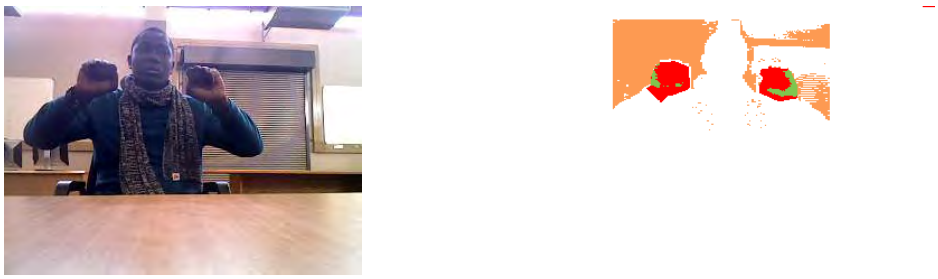


**Figure 4.7.** Incorrect classifications using the explicit skin-colour segmentation.

## 4.2. High Level Feature Extraction

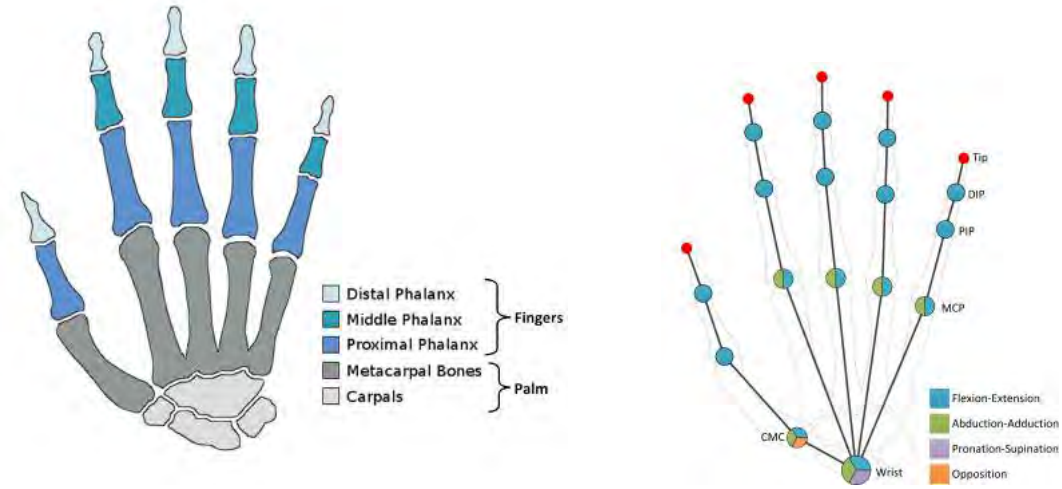
As stated in Section 2.2, there are two main approaches to gesture recognition, appearance-based and model-based methods. Appearance-based approaches are limited in that they cannot be easily extended, whereas model-based approaches are typically resource intensive methods. However, as the requirements in Section 1.2 state, for a gesture recognition system to be successful it must be reconfigurable. Therefore, the model-based approach is ideal as it is easily extensible. This is the approach adopted in this work.

After the hand has been segmented from the image using depth thresholding and skin colour analysis, a hand model should be created. The parameters of the model will then be used to determine which gesture is being performed. The approach presented uses the hand depth image and inverse kinematics to determine the hand pose, combining the approaches of Mo and Neumann [98] and Chua et al. [99]. The depth image is used to locate seven characteristic points on the hand: the five fingertip positions, the palm centre, the wrist centre, and the forearm orientation. Using inverse kinematics the pose of the fingers are calculated using the fingertip position and finger lengths. The pose of the hand can be determined using the forearm orientation. The following steps describe the process:

- The user is asked to present their hands to the camera with the palms facing forward and the fingers spread apart.
- A calibration routine is used to calculate the finger lengths and the palm and wrist size.
- The user performs a pose.
- The palm centre and wrist positions are calculated.
- The forearm is segmented from the hand, and the forearm orientation is calculated as described in Section 4.1.2.
- The hand depth image is used to locate fingertip positions.
- Inverse kinematics is used to calculate the pose of each finger. These parameters describe the hand.

### 4.2.1. Hand Model

The hand model is formed by examining the skeletal anatomy of the hand as shown in Figure 4.8a.



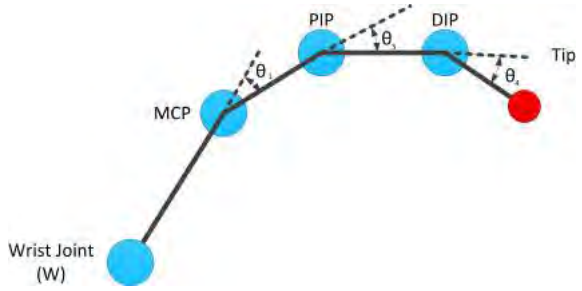

**(a)** Skeletal anatomy of the hand. **(b)** DOF of the hand model.

**Figure 4.8.** Skeletal anatomy and model of the hand.

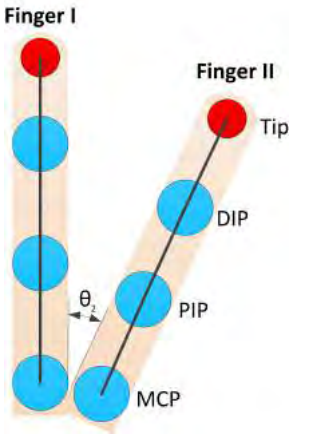
The bones of the hand are naturally divided into two groups: the thirteen bones which make up the palm; and the digits, each consisting of three phalangeal segments (except for the thumb that consists of only two phalangeal bones). The proximal row of the carpals articulates with the radius and ulna from the arm. There are six degrees of freedom (DOF) possible at this joint, namely, flexion-extension, abduction-adduction and pronation-supination, as well as translation in the $x$, $y$ and $z$ directions. These terms are explained in Appendix A. For each of the digits the functional anatomy is the same, with the exception of the thumb. The metacarpals articulate with both the distal row of the carpals and the proximal phalanges. The metacarpal-phalangeal (MCP) joint has two DOF: flexion-extension and abduction-adduction. At each interphalangeal joint, i.e. the joint between the proximal and middle phalanges (PIP) and the joint between the middle and distal phalanges (DIP), there is only one DOF, namely flexion-extension. At the joint between the carpal and metacarpal of the thumb (CMC), there are three DOF: flexion-extension, abduction-adduction and opposition. Images describing the possible motions can be found in Appendix A, and Figures A.2 to A.5.

There is a total of 27 DOF for the hand. These are summarised in Figure 4.8b. However, the movements of the fingers are largely interdependent, so the number of DOF can be significantly reduced. This is achieved by imposing the following six constraints on the hand model, that capture the motion dependencies.

(Finger angles are as defined in Figure 4.9).




**(a)** Angles $\theta_1$, $\theta_2$ and $\theta_3$ between the phalanxes of the finger in the sagittal plane.




**(b)** Parameter $\theta_2$ is defined as the angle between two adjacent fingers in the coronal plane.

**Figure 4.9.** Finger angle definitions.

**Constraint 1** The angle between the distal phalanx and proximal phalanx are strongly correlated. This correlation is given by

$$\theta_4 = \frac{2}{3}\theta_3, \tag{4.12}$$

where the angles are as defined in Figure 4.9. This constraint reduces the DOF of each finger from four to three.

**Constraint 2** As observed by Kuch and Huang [100], the angles of the metacarpal joint and the proximal phalanx are directly proportional. This means that as the finger flexes towards the palm the angle of abduction decreases. More precisely, the relationship between these two joints is defined by

$$\theta_1 = k_f\theta_3, 0 \leq k_f \leq \frac{1}{2}. \tag{4.13}$$

For dynamic cases $k_f = 0.5$, but this does not hold for static analysis. Therefore, $k$ is adjusted in the range $[0, 0.5]$ as in the work by Kuch and Huang [100].

**Constraint 3** It was noted by Lee and Kunii [101] that there is little abduction-adduction possible at the metacarpal joint of the middle finger. Therefore $\theta_2 = 0$ for the MCP joint of the middle finger.

**Constraint 4** As mentioned in Section 4.2.1, the thumb has 5 DOF. However for model simplification an angle for the degree of opposition is not defined. Rather a binary variable is defined to determine if the thumb is extended across the palm or not.

**Constraint 5** Based on anatomical studies there are typical ranges of motion (ROM) defined for each angle. These are seen in Table 4.1.

**Table 4.1.** Ranges of motion of the hand joints.

| | $\theta_1$ | $\theta_2$ | $\theta_3$ | $\theta_4$ |
|---|---|---|---|---|
| Thumb | 70° | 70° | 90° | - |
| Index/Ring/Pinky | 90° | −15° to 15° | 100° | 70° |
| Middle | 90° | 0° | 100° | 70° |

In addition, limits are imposed on the length between the finger joints. Studies by Buryanov and Kotiuk [102] show that the segments of the finger are defined by the following ratio:

$$l_1 : l_2 : l_3 = 0.48 : 0.3 : 0.22 \tag{4.14}$$

where $l_1$ is the length of the proximal phalanx, $l_2$ is the length of the middle phalanx and $l_3$ is the length of the distal phalanx. Therefore given the length of the finger, the phalanx lengths can be calculated.

These constraints impose an area where the fingertip can be located. For example, consider the case where the MCP is located at position $(0, 0)$ in the sagittal plane. The valid region where the tip may occur is shown in Figure 4.10. Therefore, inverse kinematics can be used to calculate the finger joint angles given the fingertip position.

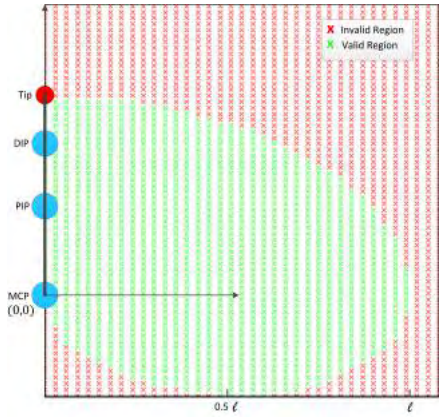



**Figure 4.10.** Valid fingertip region where the fingertip can occur if the MCP joint is located at position $(0, 0)$ in the sagittal plane. Parameter $l$ is the length of the finger.

**Constraint 6** As noted, the thumb has an additional degree of freedom allowing it to move along a non-trivial axis. However, through experimental observations done by Rijpkema and Girard [103], the kinematics of the thumb can be represented by the following two equations:

$$\theta_1 = 2\left(\theta_3 - \frac{1}{6}\pi\right) \tag{4.15}$$

$$\theta_2 = \theta_4 \frac{7}{5} \tag{4.16}$$

where $\theta_1$ and $\theta_2$ are as defined in Figure 4.11.

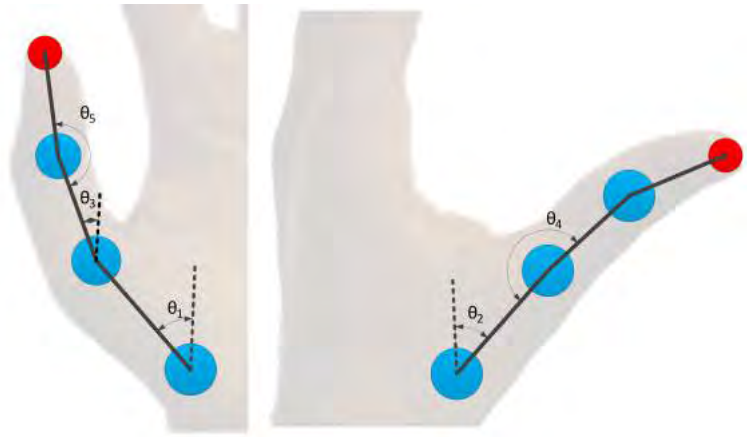



**Figure 4.11.** Angle definitions of the thumb.

## 4.2.2. Fingertip Detection

To determine the position of the fingertips an approach similar to that present by Mo and Neumann [98] is used. In this work, the authors use low resolution depth images to recognise hand poses by decomposing the problem into finger states. That is, they look at the pose of each finger (up, forward, bent, closed) and the relationship between fingers (separated, together, cross, loop).

Fingers have the following characteristics:

- They have curved/rounded tips.
- Fingertips are located at regions where there is a difference in depth.
- A finger can be decomposed into two parallel lines and a rounded tip.

Using these characteristics three methods of finding fingertips in an image are defined:

- Profile curvature-based: used for fingers that are extended.
- Depth-based: used for fingers where the fingers are flexed and the finger and palm region are merged in the hand profile.

- Parallel border-based: used when fingers are together.

Examples of each of these cases can be seen in Figure 4.12.

**(a)** Extended. **(b)** Flexed. **(c)** Together.

**Figure 4.12.** Examples of the different states that a finger can occupy.

Extended fingers can be located in the hand image using curvature based techniques. As noted, fingers typically have curved tips or peaks. Therefore, the k-curvature algorithm presented by Segen and Kumar [81] is used to identify valleys and peaks in the contour of the hand.

Given a set of points that describe the hand contour, the algorithm identifies points that occur at valleys and peaks. For each point in the set of points, the angle between the two lines that start at the point in question and end $k$ points away in either direction is calculated. If the angle is less than a threshold then the point is marked as a candidate fingertip. To delineate between valleys and peaks the distance between the points and the palm centre is calculated, and only points that are further away than the radius of the palm are kept. The approach is illustrated in Figure 4.13. The primary disadvantage of this approach is the adjustment of the parameter $k$. Depending on the size of the hand in the image, the number of points checked should differ. This is accounted for by using $k = 0.5r_p$ where $r_p$ is the radius of the palm found in Section 4.1.2. This is based on human hand proportionality studies [102]. These show that on average the radius of the palm is approximately 80% of the length of the fingers. Candidates close to one another are clustered, and the centroid is used as the fingertip candidate.

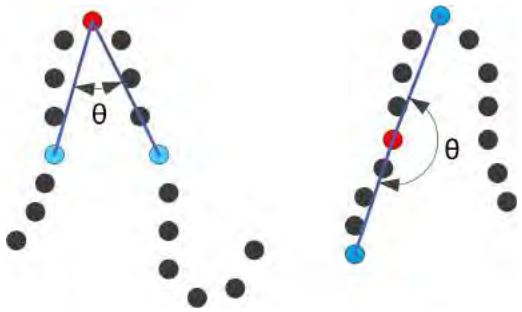



**Figure 4.13.** Illustration of the k-curvature algorithm. In the left image, the point is at a peak whereas the image on the right depicts the case where the point is not a peak/valley point.

**Tuning the Angle Threshold** The k-curvature algorithm requires two parameters, $k$ and the angle threshold $\tau$. Whilst $k$ is related to the length of the fingers, $\tau$ measures the curvature of the finger. This parameter was set using a subset of the hand gesture dataset. Using hand-labelled ground truth data the percentage of fingers correctly identified and the percentage of false positives was calculated for each image. The true positive rate (TPR) and false positive rate (FPR) as a function of $\tau$ can be seen in Figure 4.14. Ideally, the true positive rate (TPR) must be 100% and the false positive rate must be 0%. Therefore $\tau = 0.1$ is used as a higher TPR is favoured and an FPR of less than 15%.

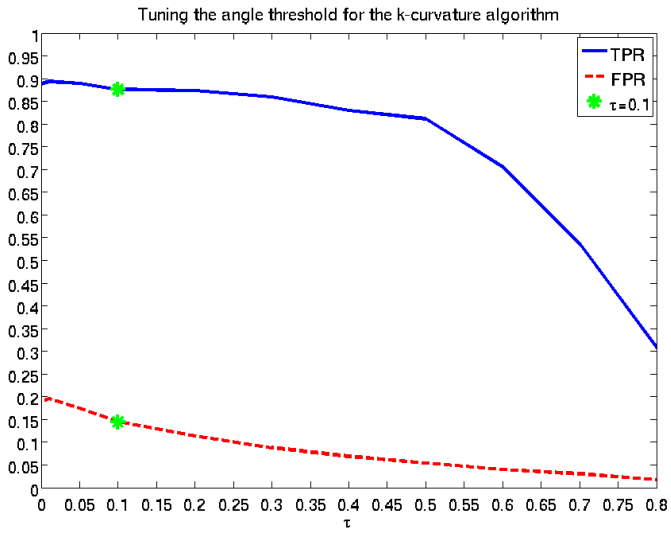



**Figure 4.14.** Tuning the angle threshold $\tau$.

If the fingers are not extended depth-based finger detection is employed. Unlike the vast majority of works, fingertips that are not extended can be detected using the approach proposed by Mo and Neumann [98]. Fingertips are always located where there is a noticeable depth difference between neighbouring pixels. Therefore, search the depth images for pixels where this occurs:

$$D = \{p | p \in H \cap \Delta\} \tag{4.17}$$

$$\Delta := (\exists q \in H, ||p - q||^2 = 1 \cap z(q) - z(p) > \delta \tag{4.18}$$

where $z(x)$ is the depth at point $x$ and $H$ is the set of points in the hand region. Curvature-based fingertip detection is then used to identify fingertips on the depth boundaries. If no fingertips are found, the parallel border-based approach is used.

The parallel-border based method uses the parallel borders of fingers to detect fingertips. The Hough line detection algorithm is used to find all parallel depth boundaries in the image. Candidate fingertips are then located between two parallel boundaries. The number of fingers is allocated based on the distance between the left and right

borders using finger widths obtained from calibration. For example, suppose that the distance between the left and right borders is 4 cm and it is known that the index and middle fingers are 2 cm, two fingers will be placed between these borders.

### 4.2.3. Hand Calibration

During hand calibration the initial position of the MCP joint is obtained, the length of the fingers and the width of the fingers.

The MCP joint is located between the metacarpal bone and the phalanges on the palm of the hand. Ideally this point is located on a line parallel to the finger boundaries, passing through the fingertip at the base of the finger. These characteristics are used to calculate the initial position of the MCP joint. The algorithm is illustrated in Figure 4.15.

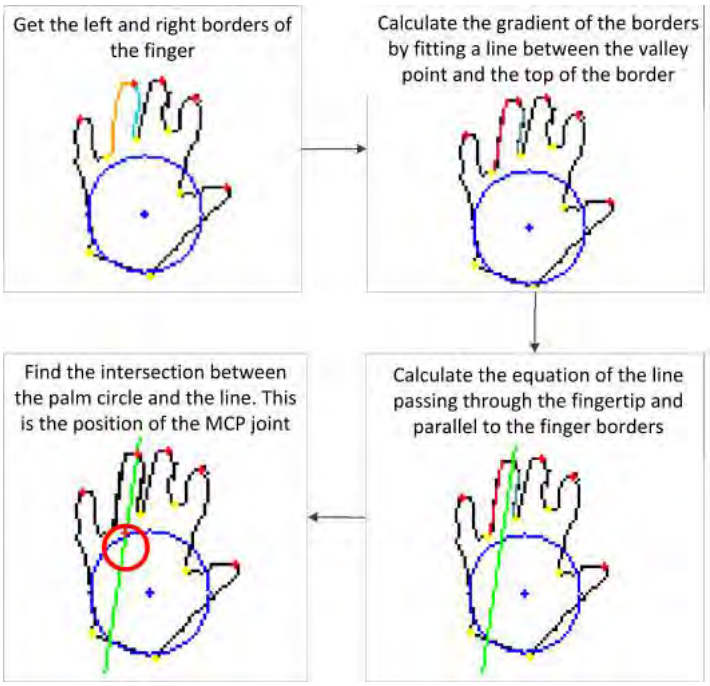



**Figure 4.15.** Calculating the initial position of the MCP joint corresponding to the second finger from the left.

The intersection between the line and the circle is calculated as shown in Appendix C. This yields two points where the line intersects the palm circle. The point with the smaller $y$ value is the MCP joint as it is assumed that the fingers are above the wrist joint. This is a valid assumption as the flexion-extension of the wrist joint is not included in the hand model.

Whilst the proportions of the finger and hand segments are robust enough for general use, it is still necessary to obtain the finger lengths and widths for each individual.

This is done using a calibration scheme depicted in Figure 4.16. The user holds up their hand with the fingers spread as far apart as possible. Curvature-based finger detection is then used to locate the five fingertips in the image. These positions are converted from the 2D image positions to 3D world coordinates using the depth image and intrinsic camera parameters. By assuming that the fingers are straight and the rotation of the hand is zero, the standard proportions provided in Section 4.2.1 can be used to calculate the 3D coordinates of the hand joints and the joint lengths. These parameters are required for the hand model. This automatic calibration routine makes the system robust over a broad range of users, even if they have not been used to train the system.

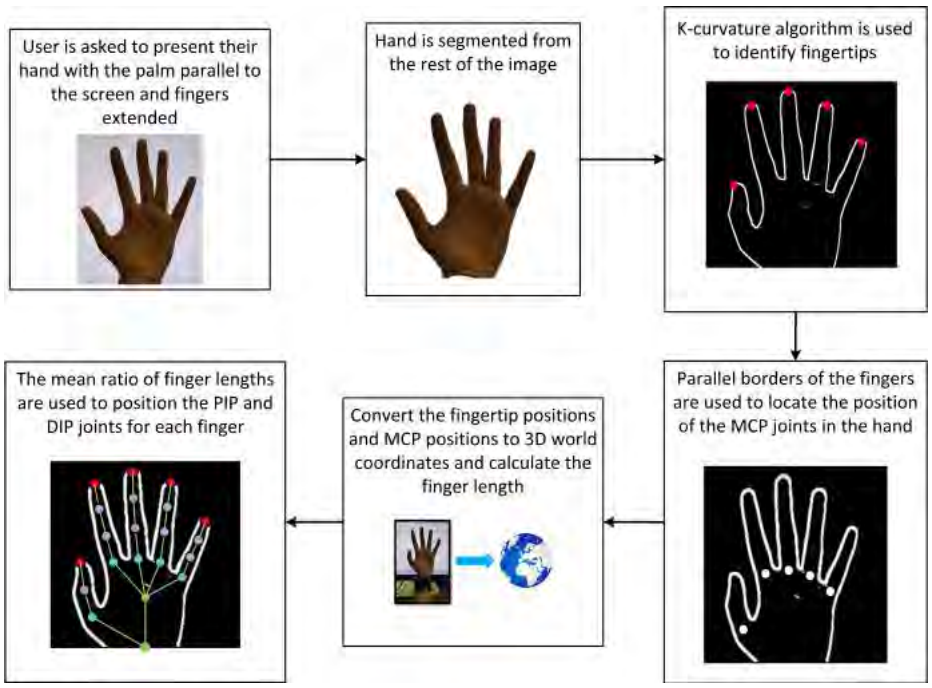



**Figure 4.16.** Method for calibrating the user hand parameters.

### 4.2.4. Joint Angles

Inverse kinematics is used to obtain the angles of the finger joints as presented by Chua et al. [99]. Consider a single finger in the sagittal plane. The distance between the tip $T$ and the wrist joint $W$, is defined as $R$. The geometry of the finger in the sagittal plane is shown in Figure 4.17 [99].

The geometry is represented by a pentagon and several triangles, therefore the following set of equations is obtained:

$$\mathrm{WP}^2 = l_0^2 + l_1^2 - 2l_0 l_1 \cos(180 - \theta_1) \tag{4.19}$$

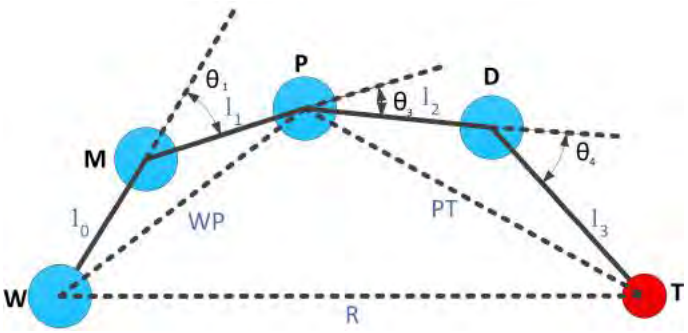



**Figure 4.17.** Geometry of the finger in the sagittal plane.

$$\mathrm{PT}^2 = l_2^2 + l_3^2 - 2l_2 l_3 \cos(180 - \theta_4) \tag{4.20}$$

$$\cos\alpha = \frac{l_1^2 + \mathrm{WP}^2 - l_0^2}{2l_1\mathrm{WP}^2} \tag{4.21}$$

$$\cos\beta = \frac{l_2^2 + \mathrm{PT}^2 - l_3^2}{2l_2\mathrm{PT}^2} \tag{4.22}$$

$$\gamma = 180 - \theta_3 - \alpha - \beta \tag{4.23}$$

$$\mathrm{R}^2 = \mathrm{WP}^2 + \mathrm{PT}^2 - 2(\mathrm{WP})(\mathrm{PT})\cos\gamma. \tag{4.24}$$

From the calibration phase of the system the finger lengths $l_0$ to $l_3$ can be obtained. In addition, once the fingertips are identified using the algorithms provided in Section 4.2.2 the 3D position of $T$ is available. It is noted by Chua et al. [99] that there is a relationship between $\theta_3$ and $R$, the distance between the wrist joint and the fingertip. This relationship is shown in Figure 4.18. Therefore, given R the value of $\theta_3$ can be found. The constraints given in Eq. (4.12) and Eq. (4.13) are then used to calculate $\theta_1$ and $\theta_4$. It was noted in Section 4.2.1 that the value of $k$ is variable. To obtain a valid solution, the value of $k$ is varied such that the finger pose is valid.

To find the value of $\theta_2$ it is assumed that this angle can be represented by a single link between the MCP joint and the tip as in the work by Chua et al. [99]. Therefore, given the fingertip position and the MCP position from calibration this value can be calculated.

Given the position of the fingertips in 3D world coordinates, and the finger joint lengths obtained from the calibration module, the joint angles of the fingers that uniquely represent the pose of the hand can be calculated. For hand gesture recognition a feature vector of dimension 21 is created for each image. This is used as input to the classifier discussed in Section 6.2.

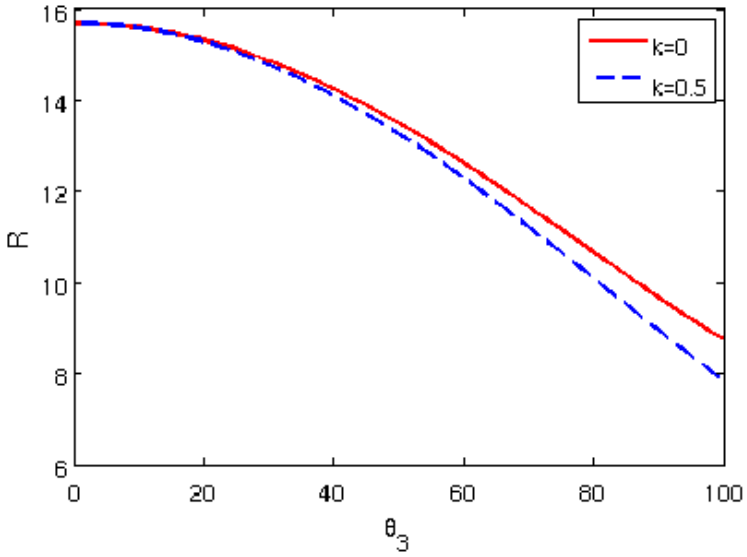



**Figure 4.18.** The relationship between R and $\theta_3$.

## 4.2.5. Results

### 4.2.5.1. Fingertip Detection

To evaluate the performance of the fingertip detection algorithms the recorded hand gesture dataset is used. Fingertips were manually marked in the colour images, red was used to mark extended fingers on the external contour of the hand and green to mark fingers that are not on the external hand contour. Samples of the ground truth images can be seen in Figure 4.19. Extended fingers are marked in red and bent or closed fingers are marked in green.

**Figure 4.19.** Sample ground truth images.

This allows us to evaluate the accuracy of the k-curvature algorithm and the depth-based fingertip detection. Two measures are used:

- The percentage of fingers identified

$$A = \frac{n_d}{n_t}, \tag{4.25}$$

where $n_d$ is the number of fingers detected and $n_t$ is the number of finger tips in the ground truth image.

- The average pixel error

$$P_e = \frac{\sum_{i=1}^{N} ||\mathbf{X}_d - \mathbf{X}_t||^2}{N} \tag{4.26}$$

where $N$ is the number of fingertips in the image, $\mathbf{X}_d$ is the location of the detected fingertip and $\mathbf{X}_t$ is the ground truth location of the fingertip.

The percentage of fingertips correctly identified can be seen in Figure 4.20.

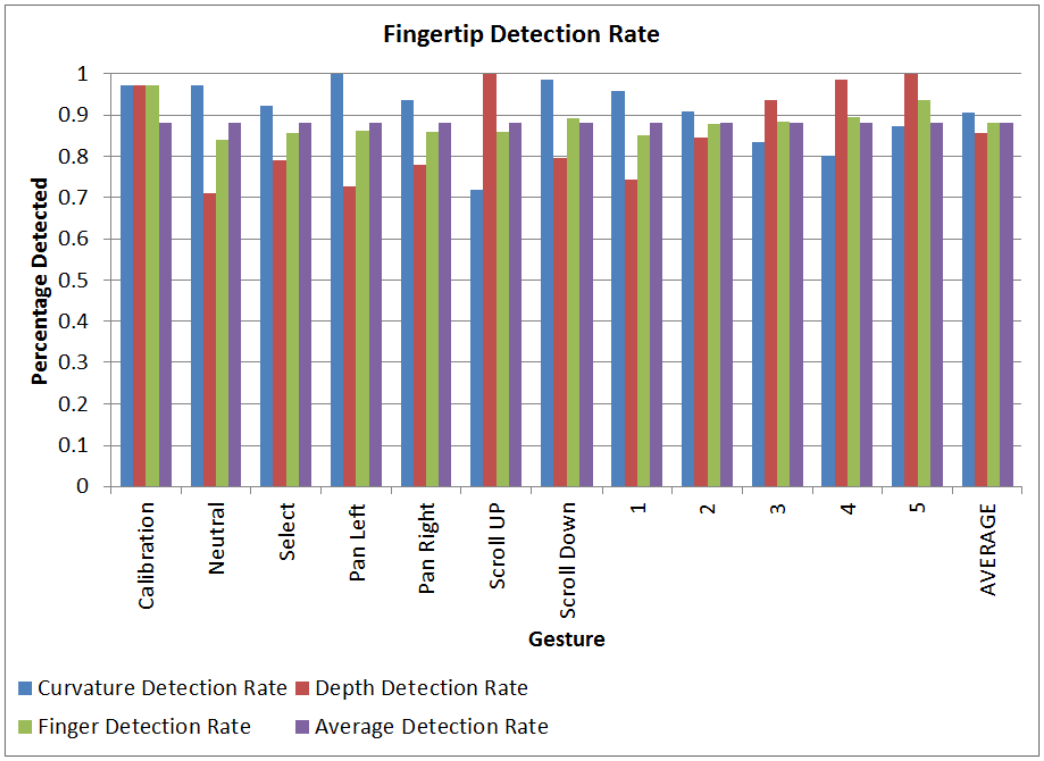



**Figure 4.20.** The percentage of fingertips identified correctly.

For extended fingers the k-curvature algorithm is robust, with only a small percentage of errors in the dataset. It can be seen that on average 88% of fingers are detected in the gesture images. The average for curvature detected fingers is 90.7% and the depth detection rate is 85.7%. This indicates that the detection methods achieve quite robust results.

The average pixel error can be seen in Figure 4.21. The error is larger for the depth

detected fingers as the internal contour is not as well defined as the external contour. Some error is expected as the ground truth fingertips were marked by hand.

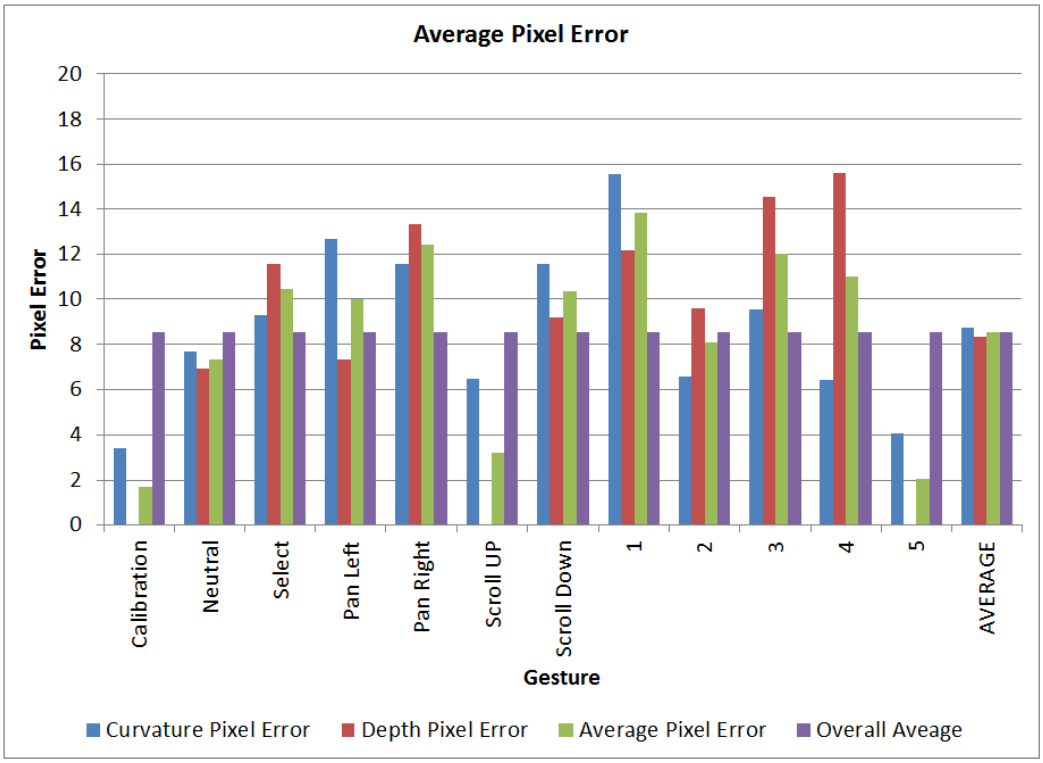



**Figure 4.21.** The average pixel error between the ground truth fingertips and the detected fingertips.

For gestures that contain primarily extended fingers, such as the *Calibration*, *Four* and *Five* gestures, the error is less than 5 pixels. A small error in this range is expected as the ground truth fingertips were marked manually. This shows that the k-curvature detection algorithm is a robust and accurate method for detecting fingers that are extended and do not coincide with the internal contour of the hand.

Gestures which contain tips in the hand contour have a larger error of approximately 9 pixels. Due the lower resolution of the depth sensor internal contours in the hand are not as smooth as the external contour, which are generated using both depth and colour. This results in more tips being detected. Fingertips are then matched to the closest MCP position and this could result in a larger error.

#### 4.2.5.2. Hand Model

The estimated hand model was qualitatively compared to the pose of the hand and number of hand models that correctly matched the gesture were counted. On average, 80.8% of hand models were correctly generated using the fingertip positions and inverse kinematics. The average accuracy is in line with the average percentage

of fingers detected, as the primary cause for incorrect hand models was incorrect fingertip detection (examples seen in Figure 4.22) and large pixel errors. Another cause of incorrect hand models is the segmentation of the hand from the forearm. In particular, the forearm of one user was not segmented from the hand, resulting in an incorrect hand model. Examples can be seen in Figure 4.23. Figure 4.24 shows samples of correctly generated hand models.

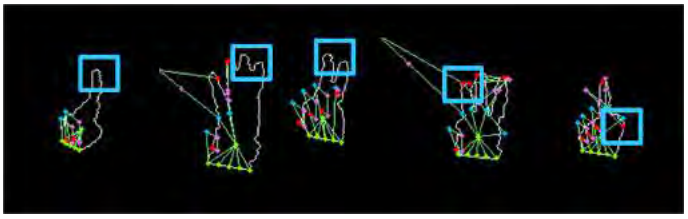


**Figure 4.22.** Samples of incorrect hand models due to incorrect fingertip detection.

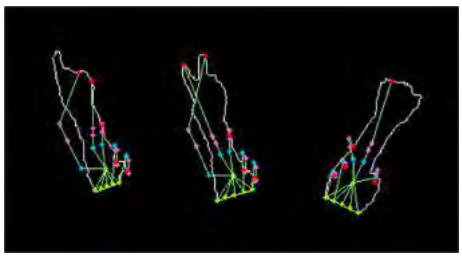


**Figure 4.23.** Samples of incorrect hand models due to incorrect segmentation.

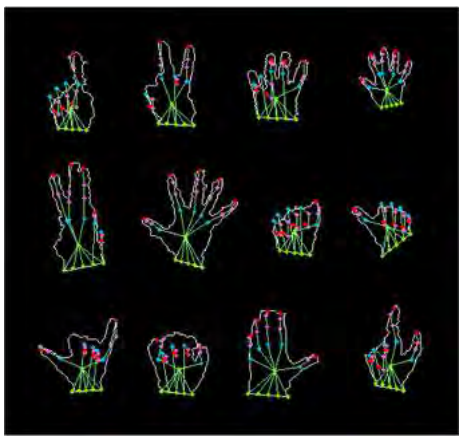


**Figure 4.24.** Samples of correctly generated hand models.

## 4.3. Summary

In this chapter the methods employed for hand gesture feature extraction were introduced. First, the hand is segmented from the image using skin colour and

depth thresholds. This approach was shown to have a small GCE of 3% compared to using only colour information where the GCE is over 14%. A novel method of calibrating the hand model for each user is presented. The fingertips are then located using the characteristics of fingers, namely the rounded tips, the difference in depth and the parallel borders. The curvature-based detection method is robust with an accuracy of approximately 91% and a small error of 5 pixels. The depth-based method achieves a fingertip detection accuracy of 86%, however, the error is larger due to the low resolution of the depth sensor. The locations of the fingertips are used to calculate the joint angles using inverse kinematics, that impose a number of constraints on valid hand poses. As opposed to other approaches that count the number of extended fingers, this method does not limit the gesture lexicon size. On average 80.8% of hand models were correctly generated. Higher results can be achieved if a higher resolution depth sensor is used or the methods of detecting fingertips are improved.

# 5. Body Gesture Feature Extraction

Real-world gesture recognition involves users who are untrained in gesture performance, resulting in data that is noisy in terms of gesture starts, gesture trajectory and gesture ends. For this reason, focus is placed on upper body gestures where the gesture duration and side of the body used are not prescribed. In such cases the features extracted from the non-gesturing hand may skew the results as each user may not keep this hand in the same position, resulting in poor classification accuracy.

The approach used to address the noisy gesture trajectories is presented in this chapter. First joint angles are extracted using joint positions from the NiTE skeleton as described in Section 5.2. The joint angles are used as an input to the classifier for upper body gesture recognition. The velocity-based feature vector is described in Section 5.3. This feature vector is used to determine which side of the body is being used for gesture performance. Finally, key-frame selection using the euclidean distance metric and dynamic time warping are presented in Section 5.4. These algorithms are used to temporally align the gesture trajectories, ensuring that all trajectories have the same duration. The results of the temporal alignment are reported in Section 5.4.3.

## 5.1. Data Collection

The Asus Xtion Pro Live depth sensor generates depth maps, RGB images and audio streams without requiring the user to wear any additional aids [104]. To interface with the device, the OpenNI [105] and NiTE Software Development Kits (SDKs) [89] are used. These SDKs provide a control API to the end user by utilising the depth, RGB and audio information received from the depth sensor. In particular, the NiTE library includes a skeleton tracker [89]. The skeleton model generated by NiTE is a tree graph whose nodes correspond to individual joints in the human body, as seen in Figure 5.1. The skeleton tracker tracks the 3D $(x, y, z)$ coordinates of these 15 joints in real-time, at 30 fps. The skeleton model is robust to differences in user size and shape, clothing colour and texture and background clutter, making it ideal for feature generation.

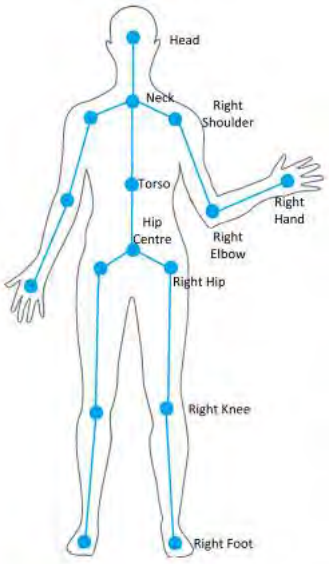



**Figure 5.1.** NiTE Skeleton.

## 5.2. Joint Angles

This work only considers upper body gestures. Therefore only the joints in the upper body are of interest, namely the left shoulder (ls), right shoulder (rs), left elbow (le), right elbow (re), left hand (lh), right hand (rh) and head (he) joints.

### 5.2.1. Joint Angles

The distance between joints is affected by the height of the user and the distance from the camera. Therefore, relative distance is not a scale invariant feature. Joint angles on the other hand are both scale and rotation invariant, as they are not dependent on the height of the subject, the distance from the camera, or the orientation of the user relative to the camera plane. Six joint angles were calculated for each pose. These are shown in Figure 5.2 [106].

Figure 5.3 illustrates the calculation of the elbow angle.

To calculate the joint angle the vector between joints must be computed. The shoulder-elbow vector $\overline{s-e}$ and elbow-hand $\overline{e-h}$ vector are given by Eq. (5.1) and Eq. (5.2) respectively:

$$\overline{s-e} = (x_2 - x_1)\hat{i} + (y_2 - y_1)\hat{j} + (z_2 - z_1)\hat{k}. \tag{5.1}$$

$$\overline{e-h} = (x_2 - x_3)\hat{i} + (y_2 - y_3)\hat{j} + (z_2 - z_3)\hat{k}. \tag{5.2}$$

The coordinates $(x, y, z)$ are as defined in Figure 5.3.

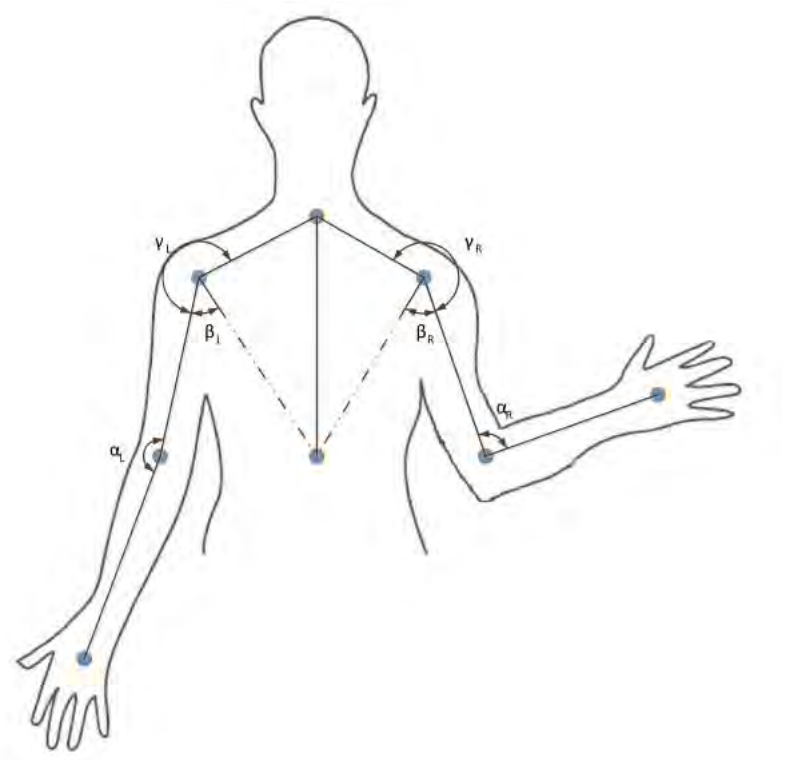



**Figure 5.2.** Joint angle calculated for the upper body gesture recognition feature vector .

The elbow angle is then given by

$$\theta = \arccos\left(\frac{\overline{s-e}\cdot\overline{e-h}}{|\overline{s-e}||\overline{e-h}|}\right), \tag{5.3}$$

where the numerator $\overline{s-e}\cdot\overline{e-h}$ is the scalar product of the corresponding vectors. The denominator is a normalising factor such that the scalar product is of unit length.

The six-dimensional joint angle feature vector is defined as follows:

$$F_{JA} = [\gamma_L, \gamma_R, \beta_L, \beta_R, \alpha_L, \alpha_R], \tag{5.4}$$

where the symbols are as shown in Figure 5.2.

### 5.2.2. Relative Joint Positions

As joint angles are rotation invariant, a pose with the arms stretched on either side of the torso and arms stretched in front of the torso will have similar feature vectors. Therefore, the relative joint position between the elbow and hand joints and the head joint is calculated for each pose. Figure 5.4 shows the position of the hand

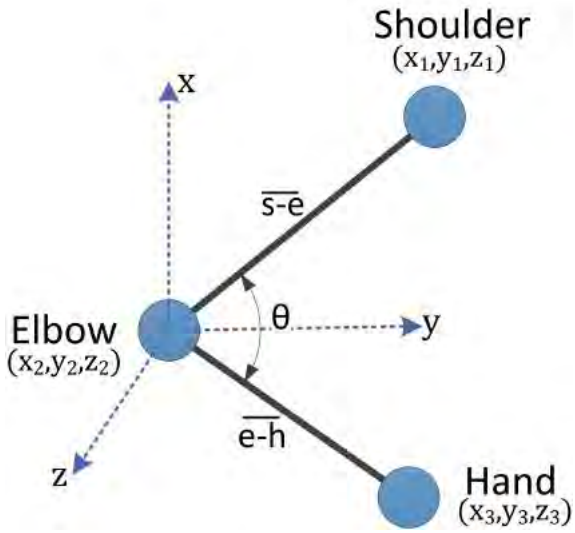



**Figure 5.3.** Calculation of the elbow angle.

relative to the $x$-component of the head joint. Similarly to Eq. (5.1) the head-hand vector is given by

$$\overline{he-h} = (x_2 - x_1)\hat{i} + (y_2 - y_1)\hat{j} + (z_2 - z_1)\hat{k}. \tag{5.5}$$

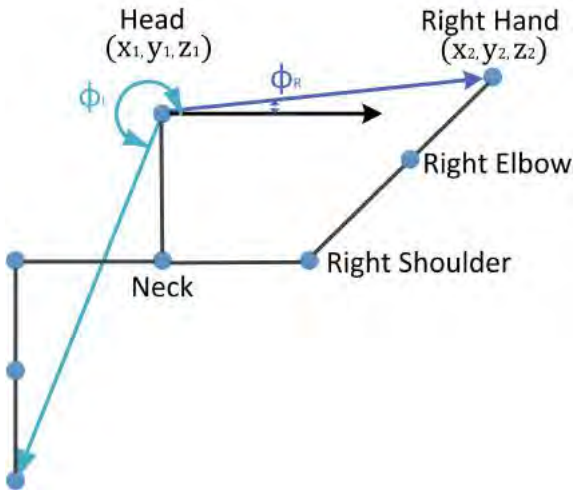



**Figure 5.4.** Relative position of the hand and elbow with respect to the head.

The $x$-component of the head joint is

$$he_x = x_1\hat{i} + 0\hat{j} + 0\hat{k}. \tag{5.6}$$

Thus the position of the hand relative to the head is

$$\varphi = \arccos\left(\frac{\overline{he-h}\cdot\overline{he_x}}{|\overline{he-h}||\overline{he_x}|}\right). \tag{5.7}$$

If the joint is below the head, the angle is subtracted from 360° to ensure that relative joint positions have unique angular representations.

### 5.2.3. Combined Feature Vector for upper body Gesture Recognition

The joint angles, relative joint positions and the distance between the left and right hands are combined to form an eleven-dimensional feature vector

$$F_C = [\gamma_L, \gamma_R, \beta_L, \beta_R, \alpha_L, \alpha_R, \varphi_L, \varphi_R, \sigma_L, \sigma_R, lh - rh]. \tag{5.8}$$

Table 5.1 on page 51 provides a brief description of each element this feature vector.

**Table 5.1.** Description of the elements in the feature vector used for upper body gesture recognition.

| | | | |
|---|---|---|---|
| $\gamma$ | Elbow-Shoulder-Neck angle | $\varphi$ | Relative position of the elbow relative to the head |
| $\beta$ | Torso-Shoulder-Neck angle | $\sigma$ | Relative position of the hand relative to the head |
| $\alpha$ | Hand-Elbow-Shoulder angle | $lr-rh$ | Distance between the left and right hands |

Therefore, for each frame in the gesture sequence an eleven–dimensional feature vector is calculated. This feature vector is used for training and testing.

## 5.3. Gesture Spotting

A common problem in temporal gesture recognition is gesture spotting. Gesture spotting locates a gesture in a sequence of signals. An approach employed by Cheng et al. [107] uses the acceleration signal of the hand to determine if a gesture is being performed. There are three distinct phases in a gesture signal: a high-speed start, a continuous change in direction and an end in an almost steady position. In order to identify the gesturing side a similar approach is employed. The velocity of the left and right hands are calculated as

$$v = \frac{\sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2 + (z_t - z_{t-1})^2}}{fps^{-1}}. \tag{5.9}$$

Like the acceleration signal, the velocity signal of the gesturing hand will show a steady increase whilst that of the non-gesturing hand will be largely stationary. This relationship is illustrated in Figure 5.5.

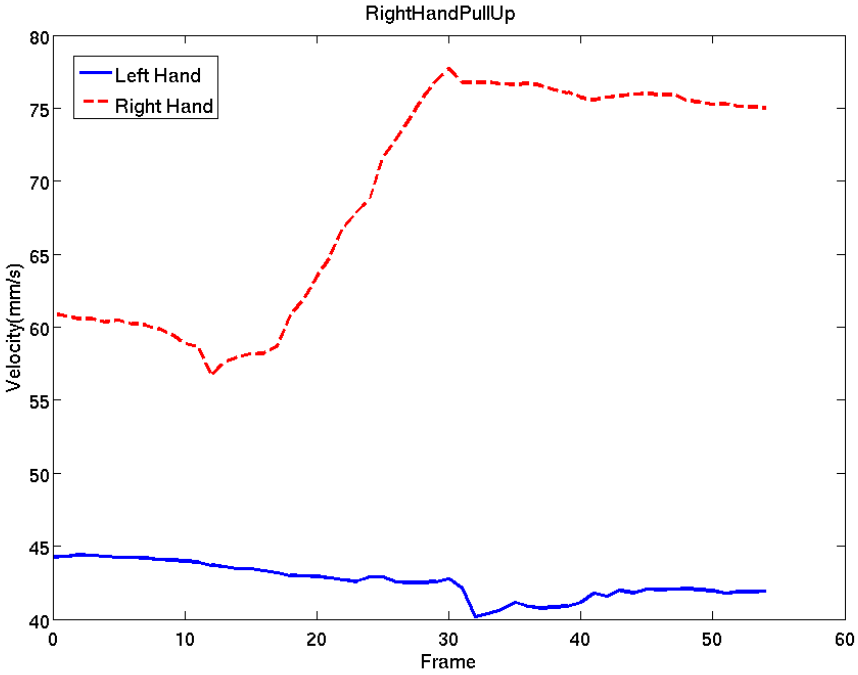



**Figure 5.5.** Velocity profile of the left and right hands when the "Right Hand Pull Up" gesture from [28] is performed.

It is evident that the velocity profiles of the gesturing and non-gesturing hands are distinct from one another. The velocity of the non-gesturing hand is steady throughout the gesture whereas the gesturing hand has some variable velocity trajectory. To capture this characteristic the tenth percentile and ninetieth percentile values of the velocities are found. The difference between these values is used to distinguish between left and right side gestures. It is expected that the gesturing hand will show a much greater difference between these velocities compared to the non-gesturing hand. This difference is calculated as

$$side\ feature = v(i) - v(j), \tag{5.10}$$

where $i$ is the index of the ninetieth percentile and $j$ is the index of the tenth percentile. The velocity signal is calculated for the duration of the gesture. This calculation is possible as the gestures in the datasets are already segmented, but they are not temporally aligned.

In addition, the difference between the ninetieth percentile and the value at the mid-point between the tenth and ninetieth percentile values is calculated. This value is used to capture any sudden changes in gradient that may be present and is

calculated as shown

$$x = v\left(\frac{i-j}{2}\right), \tag{5.11}$$

where $v(\cdot)$ is the velocity of the hand and $\frac{i-j}{2}$ is the mid-point between the tenth and ninetieth percentile values. i.e. Eq. (5.11) is the velocity of the hand at the mid-point.

## 5.4. Temporal Alignment of Gesture Trajectories

Real world data collected from untrained users often contains trajectories that are temporally misaligned. This misalignment is caused by gestures not starting at the same time, having different durations and ending at different times. Efforts to recognise gestures will give inaccurate results if there is no temporal alignment. Therefore the data must be temporally aligned. Two methods are considered — the dynamic time warping (DTW) algorithm that compares the gesture sequence to a template and the novel key-frame selection algorithm that selects only frames where there is significant motion.

### 5.4.1. Dynamic Time Warping

DTW is a well-known technique for finding the optimal temporal alignment between two time-dependent signals [108]. Consider two sequences $X := (x_1, x_2, ..., x_N)$ of length $N \in \mathbb{N}$ and $Y := (y_1, y_2, ..., y_M)$ of length $M \in \mathbb{N}$. The points in the sequence must be equidistant in time. To compare two different features $x_n$ and $y_m$ a local cost corresponding to the distance between points $x_n$ and $y_m$ must be defined. An $n \times m$ cost matrix consisting of the cost distance between the points is defined. The goal of DTW is to find that path through the matrix that minimises the total cumulative distance between them. The optimal path that minimises the warping cost is

$$DTW(X,Y) = \min\left\{\sqrt{\sum_{k=1}^{K} w_k}\right\}, \tag{5.12}$$

where $w_k$ is the matrix element $(n,m)_k$ that belongs to the $k^{\text{th}}$ element of a warping path $W$, a contiguous set of matrix elements that represents a mapping between $X$ and $Y$. The warping path $W$ can be found using dynamic programming to evaluate

$$\delta(n,m) = d(x_n, y_m) + \min\left\{\delta(n-1, m-1), \delta(n-1, m), \delta(n, m-1)\right\}, \tag{5.13}$$

where $d(n,m)$ is the distance in the current cell, and $\delta(n,m)$ is the cumulative distance of the $d(n,m)$ and the minimum cumulative distances from three adjacent cells.

Three constraints are introduced to limit the search space during computation [108]:

- *Boundary conditions:* the first and last elements of $X$ and $Y$ must be aligned to one another. Mathematically, $w_1 = (1, 1)$ and $w_K = (N, M)$.
- *Monotonicity condition:* if an element in $X$ precedes a second one this should also hold for the corresponding elements in $Y$. In other words $n_1 \leq n_2 \leq ... \leq n_K$ and $m_1 \leq m_2 \leq ... \leq m_K$.
- *Step size condition:* this ensures that no element in $X$ and $Y$ can be omitted and there are no replications in the alignment, so $w_{k+1} - w_k \in \{(1, 0), (0, 1), (1, 1)\}$ for $k \in [1 : K - 1]$.

However, DTW has a number of disadvantages:

- A template sequence must be provided to align the sequence. As it is not known which gesture is being performed the sequence will have to be aligned to all gesture templates increasing the computational requirements.
- Traditional DTW is computationally expensive, particularly for problems with many sequences [109].

Therefore, rather than traditional DTW, key-frame selection is used to align sequences temporally and ensure that they are all the same length.

### 5.4.2. Key-Frame Selection

The novel key-frame selection algorithm to temporally align gesture sequences is presented. These key-frames are selected on the basis that there must be significant movement between consecutive frames:

- *Step 1*: The Euclidean distance between feature vectors in successive frames is calculated.
- *Step 2*: If the distance is below the threshold the corresponding point is discarded. The threshold value was found empirically to be 0.1.
- *Step 3*: Using the remaining points, the minimum sequence length is found.
- *Step 4*: All sequences are shortened to the minimum sequence length. For example, if the minimum sequence length is ten and a sample had twenty key-frames the ten frames with the smallest distance from the previous frame are discarded.

The method is illustrated in Figure 5.6.

### 5.4.3. Results

As discussed in Section 5.4, key-frame selection was used to temporally align gesture sequences and ensure that all sequences are the same length. To evaluate the performance of the algorithm the log sum squared error (SSE) between the average

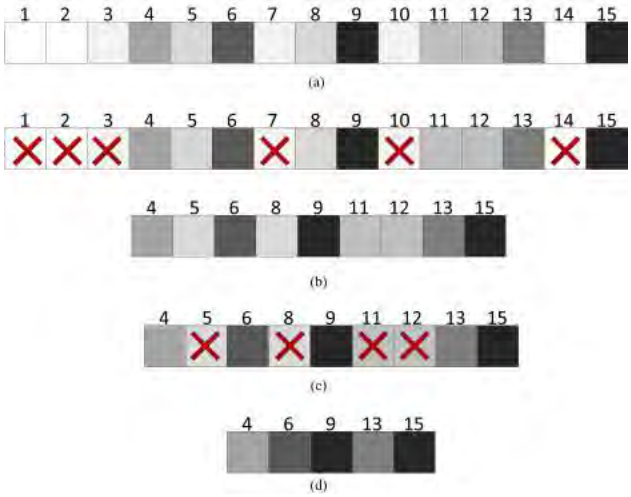


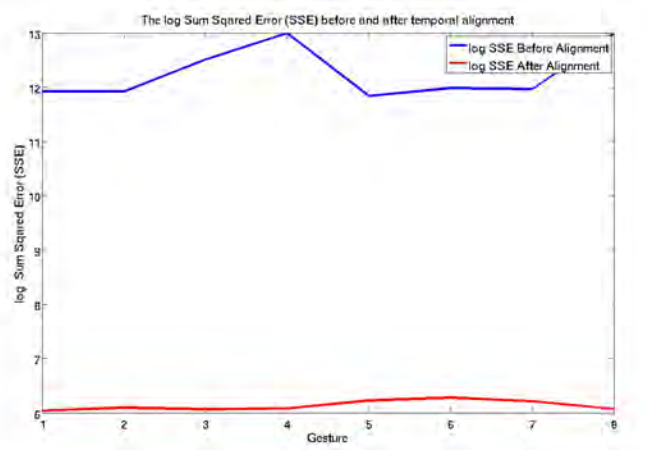

**Figure 5.6.** Illustration of the key-frame selection algorithm. The goal is to temporally align gesture sequences and ensure that all sequences are the same length. (a) Original sequence. The intensity of the block is proportional to the distance between the feature vectors from the previous and current frame. Solid black indicates maximum distance. (b) Frames with a distance less than the threshold are discarded. (c) To obtain sequences of the same length frames with the minimum distance are discarded. (d) Final sequence.

sample in each gesture set and each sample in the dataset is calculated. These are shown in Figure 5.7.

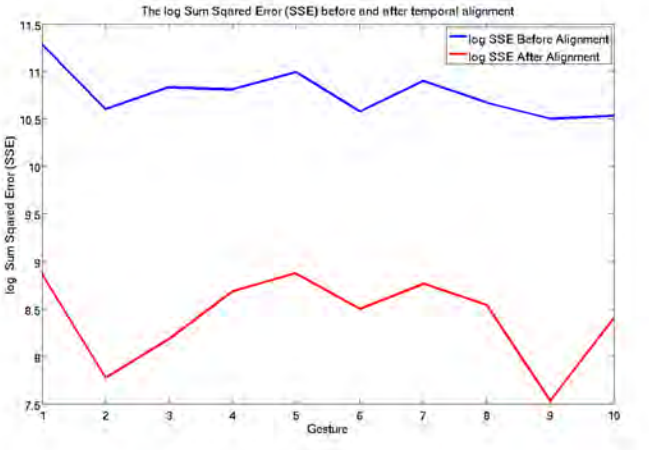
The log squared error is reduced by at least one order of magnitude in both the VisApp2013 and recorded datasets implying that the key-frame selection improves the alignment between gesture sequences. Qualitative results showing the alignment between two gesture trajectories can be seen in Figure 5.8. Before alignment the gesture sequences start at different times and have different lengths. After key-frame selection the sequences are the same length and start at the same time.

## 5.5. Summary

This chapter describes the features extracted for upper body gesture recognition. A feature vector consisting of joint angles and relative joint positions is extracted from the NiTE skeleton. These features are extracted from each frame in the gesture sequence and concatenated to form a feature vector that can be used for gesture classification. The use of the velocity of the hands is proposed as a feature for determining which side of the body is being used. In addition, a novel key-frame selection algorithm is proposed to temporally align gesture sequences. This method is chosen over DTW as it does not rely on a template gesture. The results show that the key-frame selection method is able to reduce the log squared error by at



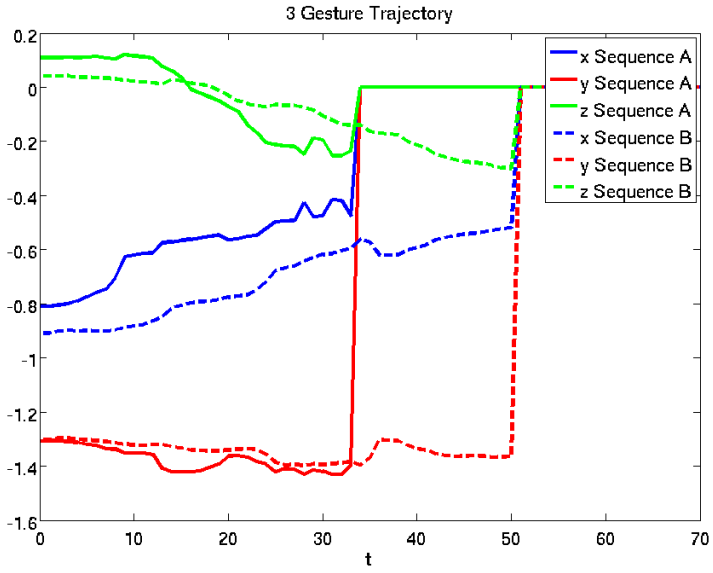

**(a)** log SSE of the VisApp dataset.



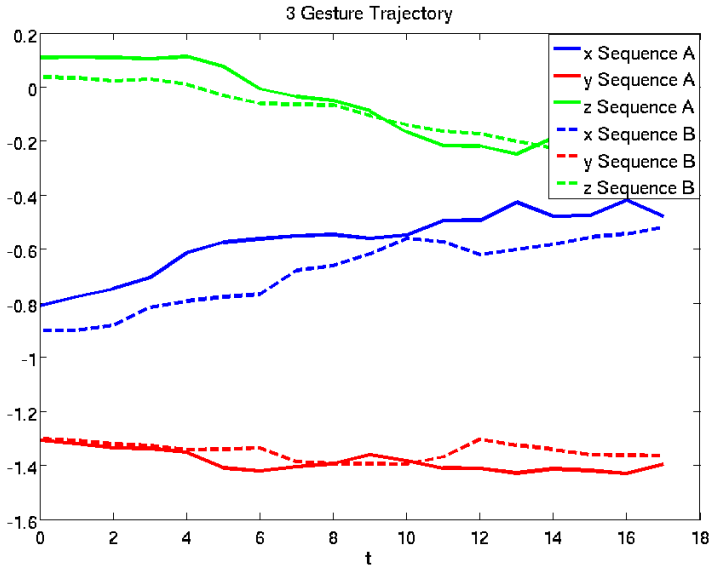

**(b)** log SSE of the recorded dataset.

**Figure 5.7.** log SSE between the average trajectory of each gesture and each gesture sample before and after alignment.

least one order of magnitude after alignment. Qualitative results indicate that the method works for temporally aligning gesture sequences.




**(a)** Before Alignment.




**(b)** After Alignment.

**Figure 5.8.** Alignment between gesture sequences.

# 6. Gesture Classification

Features extracted from the raw data are used as inputs to a classifier. The classifier is required to identify the most probable class for an unknown pattern or sequence.

Given an unknown pattern or gesture, represented by the feature vector $\mathbf{x}$, which must be classified into one of $M$ classes. The gesture is then classified by maximising the probability that an unknown pattern belongs to a given class. Mathematically

$$c^* = \arg\max_i P(c^i|\mathbf{x}), \tag{6.1}$$

where $c$ is the gesture label and $c^*$ denotes the optimal class.

This chapter presents three types of classifiers used for gesture recognition, namely are neural networks discussed in Section 6.1.1, hidden Markov models presented in Section 6.1.2 and $k$-means described in Section 6.2.1. These classifiers are used to obtain the class labels of a gesture given the feature vectors discussed in Chapter 4 and Chapter 5. The goal of each classifier is to have a large number of true positives, that is, gestures that are correctly labelled, and as few as possible misclassification's. The classifiers are evaluated in terms of their accuracy, precision and recall rates. The results of the body gesture recognition and hand gesture recognition is presented in Sections Section 6.1.3 and Section 6.2.3 respectively.

## 6.1. Body Gesture Classification

The joint angle and relative joint position features extracted from the NiTE skeleton are used as feature vectors for upper body gesture recognition as shown in Chapter 5. Two classifiers, neural networks and hidden Markov models, are compared. These classifiers are chosen as they can model temporal signals. This section discusses the theory and implementation details for the classifiers.

### 6.1.1. Neural Networks

Artificial neural networks (ANN) are widely used architectures for machine learning applications. Developed in the 1960s to try and mimic the brain, neural networks are now a state-of-the-art technique used in a variety of applications. One of the main advantages of neural networks is that multiple layers can easily model non-linearities in the data, without these relationships being pre-specified by the system

designer. They rely on the principle of "divide and conquer" where a large, complex problem can be decomposed into simpler, interconnected elements known as neurons or nodes [110]. The nodes are computational elements that receive inputs and process them to produce an output. The connections between nodes determine the flow of information from one node to another. A basic neural network can be seen in Figure 6.1.

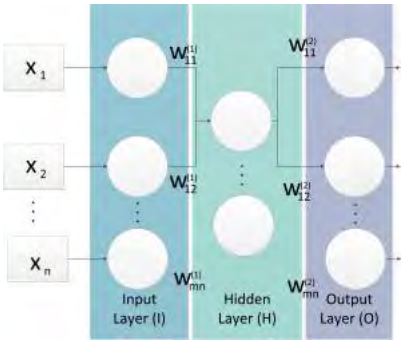



**Figure 6.1.** Basic neural network containing three layers.

Similarly to the behaviour of a neuron in the brain, when the total received signal is larger than a certain threshold the neuron is activated, and a signal is propagated to the connected neurons. By tuning or training the weights $w$ the required output for a given set of inputs can be obtained. To adjust the weights of the neural network the back propagation algorithm is used. In back propagation, inputs are fed forward whilst errors are propagated backwards, with the goal of minimising the training error [111]. The activation function is given by

$$a_j^{(l)} = \sum_{i=1}^{d} w_{ji}^{(l)} x_i, \quad j = 1 \dots M. \tag{6.2}$$

The activation of the hidden neuron is dependent on the inputs $x_i$ and the weights associated with the $l^{\text{th}}$ layer, $w_{ji}^{(1)}.a_j^{(l)}$ is the activation of the $j^{\text{th}}$ neuron in the $l^{\text{th}}$ layer. The output function is the sigmoid function by

$$O(a_j) = \frac{1}{1 + e^{a_j}}. \tag{6.3}$$

This function will be close to one for large positive numbers, 0.5 at zero and almost zero at large negative numbers. As discussed above, the goal of back propagation is to find the weights that result in the desired output for a specific input, or to minimise the error between the desired output and the actual output. Mathematically,

$$E = \sum_{j=1}^{N} (O(a_j) - d_j)^2, \tag{6.4}$$

where $d_j$ is the desired output at the $j^{\text{th}}$ neuron in the output layer. To minimise

the error, the derivative of the error function with respect to the weight parameters is evaluated.

The error gradient for each output neuron $O_k$ is given by

$$\delta_k = O_k(1 - O_k)(d_k - O_k). \tag{6.5}$$

The error gradient for each node in the hidden layer is given by

$$\delta_j = y_j(1 - y_j) \sum_{k=1}^{n} w_{jk}\delta_k. \tag{6.6}$$

This is dependent on the error in the output layer.

Finally, to update the weights

$$\begin{aligned} w_{ij} &= w_{ij} + \alpha.A_i.\delta_j \quad \text{and} \\ w_{jk} &= w_{jk} + \alpha.H_j.\delta_k, \end{aligned} \tag{6.7}$$

where $\alpha$ is the learning rate and affects the learning speed of the network, $A_i$ is the $i^{\text{th}}$ neuron in the input layer and $H_j$ is the $j^{\text{th}}$ neuron in the hidden layer.

Mini-batch stochastic gradient descent (MBSGD) is used to optimise the network weights. MBSGD is similar to gradient descent with several modifications:

- Weights are updated randomly with a chosen subset of $b$ samples of the training set, rather than a single sample at a time, to escape local minima .
- Momentum is used to smooth the search direction.
- Each weight has an adaptive learning rate.
- The learning rate and momentum can be adjusted during optimisation.

This results in a faster optimisation and discourages convergence to local minima. The OpenANN [112] implementation is used to train the neural networks in our work.

Two neural networks are used in this work, one to distinguish the gesturing side and another to determine the upper body gesture label.

A fully-connected feed-forward neural network is used to determine which side of the body is being to perform gestures. The network structure can be seen in Figure 6.2. There are four input neurons, five hidden neurons and two output neurons. The inputs are the hand velocities described in Section 5.3. The number of hidden neurons in calculated according to

$$h = \left\| \frac{2}{3}i + n \right\|, \tag{6.8}$$

here $h$ is the number of hidden neurons, $i$ is the number of input neurons and $n$ is the number of labels or outputs.

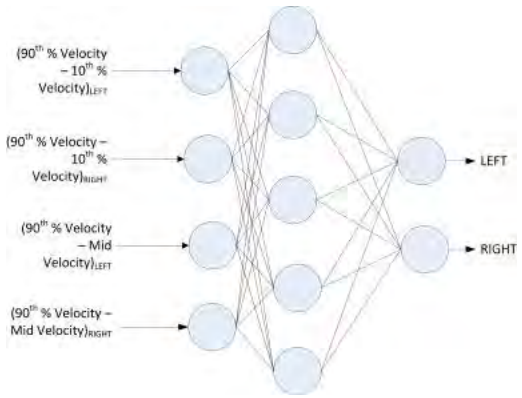



**Figure 6.2.** Neural network structure used to determine the gesture side using feature defined in Section 5.3.

A feed-forward fully-connected neural network is used for gesture classification. The network architecture is similar to Figure 6.2 except that there are 198 input neurons, 142 hidden neurons and 10 output neurons. The input neurons correspond to the eleven-dimensional feature vector presented in Section 5.2. The number of hidden neurons is calculated using Eq. (6.8) and there is an output neuron for each gesture in the lexicon.

### 6.1.2. Hidden Markov Models

Hidden Markov models (HMM) are temporal models that describe the state of a process by the evolution of a single discrete random variable. The possible values of the state are equal to the possible values of the random variable. They are used to represent probability distributions over a sequence of observations. HMMs satisfy two properties:

- An observation $\mathbf{O}_t$ at time $t$ is generated by some process whose state $\mathbf{X}_t$ is hidden from the observer. The observation,$\mathbf{O}_t$ is one of the $L$ possible observation symbols, $\mathbf{O}_t \in \{o_1 ... o_L\}$.
- Given the values of the previous state, $\mathbf{X}_{t-1}$, the current state $\mathbf{X}_t$ is independent of all states prior to $t-1$. The state $\mathbf{X}_t$ is a discrete random variable with $N$ possible values $\{1 \dots N\}$.

The joint distribution over all the variables is given by

$$\mathbf{P}(\mathbf{X}_{0:t}, \mathbf{O}_{1:t}) = \mathbf{P}(\mathbf{X}_0) \prod_{i=0}^{t} \mathbf{P}(\mathbf{X}_i|\mathbf{X}_{i-1}) \mathbf{P}(\mathbf{O}_i|\mathbf{X}_i), \tag{6.9}$$

where $\mathbf{P}(\mathbf{X}_0)$ is the initial state model representing the prior probability, $\mathbf{P}(\mathbf{X}_i|\mathbf{X}_{i-1})$ is the state transition model and $\mathbf{P}(\mathbf{O}_i|\mathbf{X}_i)$ is the sensor model.

By assuming that the states are time-independent the transition model can be represented using a stochastic transition matrix $T = \{t_{ij}\} = \mathbf{P}(\mathbf{X}_t = j|\mathbf{X}_{t-1} = i)$. When $t = 0$, the initial state model is$\pi$.

The sensor model can be described using an emission matrix $E$. The probability of a particular observation at time $t$ for state $j$ is described by $e_j(o_t) = \mathbf{P}(\mathbf{O}_t = o_t|\mathbf{X}_t = j)$. The emission matrix $E$ is an $L \times N$ matrix. Therefore,an HMM can be described by $\lambda = (T, E, \pi)$.

The transition and emission matrices are most commonly estimated from data using the Baum-Welch algorithm. Inference provides an estimate of what transitions occurred and the states that generated the sensor readings. These are used to update the models.

The Baum-Welch algorithm, Algorithm 6.1 [110], is a special case of the expectation-maximisation (EM) algorithm that is used to learn the transition and sensor models of a HMM from data. The algorithm finds the HMM $\lambda$ that maximises the probability of the observation $\mathbf{O}$. First, the transition, emission and initial state models are initialised with random initial conditions to obtain the HMM $\lambda_0$. The probability of being in state $i$ at time $t$ given the observations and $\lambda_0$ is then calculated. The probability of transitioning to state $j$ at time $t + 1$ is also calculated. These values are used to update the parameters of the HMM by summing over all time. This process is repeated until convergence.

**Application of HMMs to Gesture Recognition** Gesture recognition shares a number of similarities with other recognition tasks such as speech recognition, where HMMs have also successfully been employed. Consider a gesture with one of the trajectories shown in Figure 6.3.

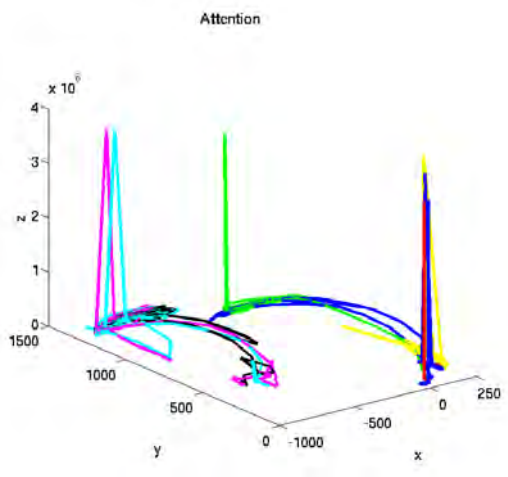



**Figure 6.3.** Gesture trajectory of seven samples of the "Attention" gesture. Each colour represents a different user.

**Algorithm 6.1** Baum-Welch algorithm

**function** BAUM-WELCH($\mathbf{X}, N, L$) **return** $\lambda = (T, E, \pi)$
  **inputs**: $\mathbf{X}$, $M \times p$ matrix of states, p sequences
    N, the number of states
    L, the number of observation symbols
  set $\lambda = (T, E, \pi)$ with random initial conditions.
  **while** not converged **do**
    **define:** $\alpha_i(t) = \mathbf{P}(\mathbf{O}_1 = o_1, ..., \mathbf{O}_t = o_t, \mathbf{X}_t = i|\lambda)$
    $\alpha_i(t) = \pi_i e_i(o_1)$
    $\alpha_j(t+1) = e_j(o_{t+1}) \sum_{i=1}^{N} \alpha_i(t) \dot{t}_{ij}$
    **define:** $\beta_i(t) = \mathbf{P}(o_{t+1}, ..., o_T|\mathbf{X}_t = i)$
    $\beta_i(T) = 1$
    $\beta_i(t) = \sum_{j=1}^{N} \beta_j(t+1) t_{ij} e_j(o_{t+1})$
    $\gamma_i(t) = \mathbf{P}(\mathbf{X}_t = i|\mathbf{O}, \lambda) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)}$
    $\zeta_{ij}(t) = \mathbf{P}(\mathbf{X}_t = i, \mathbf{X}_{t+1} = j|\mathbf{O}, \lambda) = \frac{\alpha_i(t) t_{ij} \beta_j(t+1) e_j(o_{t+1})}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i(t) t_{ij} \beta_j(t+1) e_j(o_{t+1})}$
    **Update:**
    $\bar{\pi}_i = \gamma_i(1)$
    $\bar{t_{ij}} = \frac{\sum_{t=1}^{T-1} \zeta_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$
    $\bar{e_i(k)} = \frac{\sum_{t=1}^{T} \gamma_i(t) \forall o_t = o_k}{\sum_{t=1}^{T} \gamma_i(t)}$
  **end while**
**end function**

At each instant three values are observed, the $x, y$ and $z$ positions of the hand corresponding to a state. For a gesture of duration 2 seconds there will be a total of 60 observations (30 fps). Given a HMM for the gesture the probability that an observation sequence corresponds to the "Attention" gesture, given the 60 observations, can be calculated. That is, given a sequence of observations the sequence of states that are most likely to have generated it must be found.

A HMM requires discrete states and observations. Therefore, as the $(x, y, z)$ coordinates are continuous in space the data is clustered into 30 states using $k$-means clustering. Each point in the trajectory is then assigned to the closest cluster. An observation sequence is a series of transitions between cluster centres. Clustering is performed on the entire dataset i.e. the $x, y$ and $z$ positions of the hand for all gestures. The cluster centres can be seen in Figure 6.4. The structure of the data corresponds to the normal use of the system. Each data point represents a hand position at an instant in time whilst a gesture is being performed.

The Baum-Welch algorithm [110], given by Algorithm 6.1, is used to train a HMM for each gesture. For a new observation sequence find the model that maximises the

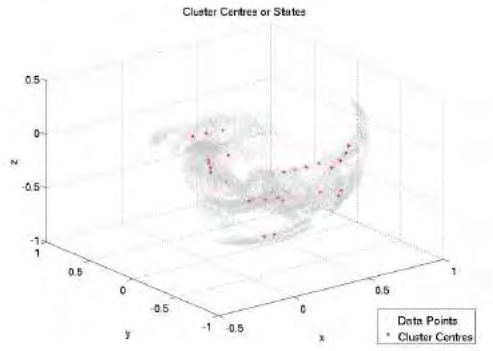



**Figure 6.4.** Cluster centres for the dynamic gesture dataset. Each point represents a state. The points in the dataset are shown in grey.

probability of the observed sequence, namely

$$c = \arg\max_i \mathbf{P}(\mathbf{O}|\lambda_i), \tag{6.10}$$

where $c$ is the gesture label, $\lambda_i$ is the HMM for the $i^{\text{th}}$ gesture and $\mathbf{P}(\mathbf{O}|\lambda)$ is the probability of the sequence given the model and is calculated using the forward algorithm given by

$$\mathbf{P}(\mathbf{O}|\lambda) = \log \sum_{i=1}^{N} \alpha_i(T), \tag{6.11}$$

where $N$ is the number of states, $\alpha_i(t)$ is the probability of seeing the partial sequence $o_1, ..., o_t$ and ending in state $i$ at time $t$, and $T$ is the time at the end of the sequence.

### 6.1.3. Results

The evaluation of the gesture recognition system will show whether the designed system meets the requirements. The classification rate is used as a performance index and is estimated by testing the response of a classifier using a finite set of $N$ test feature vectors. The total classification rate is given by

$$P_e = \frac{n_t}{N}, \tag{6.12}$$

where $n_t$ is the number of feature vectors correctly classified. In addition, the confusion matrix is used to give an indication of whether there are classes that exhibit a higher tendency for confusion. A confusion matrix $A$ is defined such that the element $A(i, j)$ corresponds to the number of data points whose true class label was $i$ and are classified to class $j$. From the confusion matrix the precision and recall values for each class can be obtained. Recall is the percentage of data points with true class label $i$ which are correctly classified in that class, i.e. the percentage

of true positives. Recall for class $i$ is given by

$$R_i = \frac{A(i,i)}{\sum_{j=1}^{M} A(i,j)}, \tag{6.13}$$

where $M$ is the number of classes in the dataset.

Precision is the percentage of data points classified as class $i$, whose true class label is $i$. For an $M$ class problem, the precision of the $i^{\text{th}}$ class is given by

$$P_i = \frac{A(i,i)}{\sum_{j=1}^{M} A(j,i)}. \tag{6.14}$$

These are used as performance measures in evaluating the gesture recognition system.

The performance of HMMs and ANNs for the purpose of gesture recognition are compared. The CMU Military dataset was used and the classification accuracy, precision and recall were calculated. Two tests were performed:

- In the first test, the dataset was split into 66.6 training data and 33.4% test data. That is, 20 samples are used for training and 10 for testing. This test is referred to as 20/10 in the results.
- In the second test the dataset were divided into 33.4% training data and 66.6% test data. This test is referred to as 10/20 in the results.

The precision and recall for the tests can be seen in Figures 6.5 and 6.6.

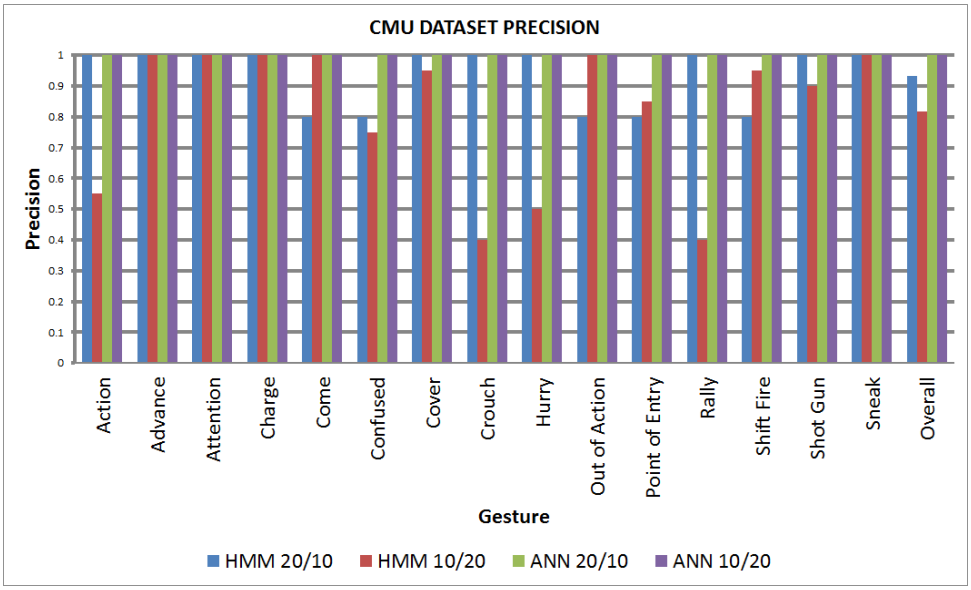



**Figure 6.5.** Precision results for the CMU dataset.

The confusion matrices for the four tests are shown in Figure 6.7.

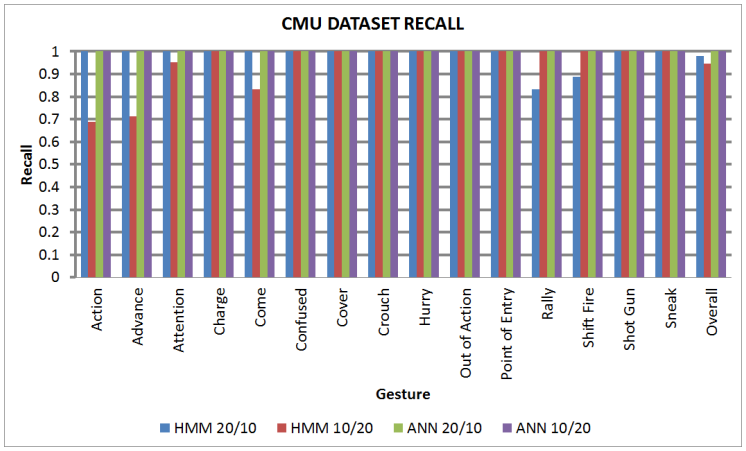



**Figure 6.6.** Recall results for the CMU dataset.

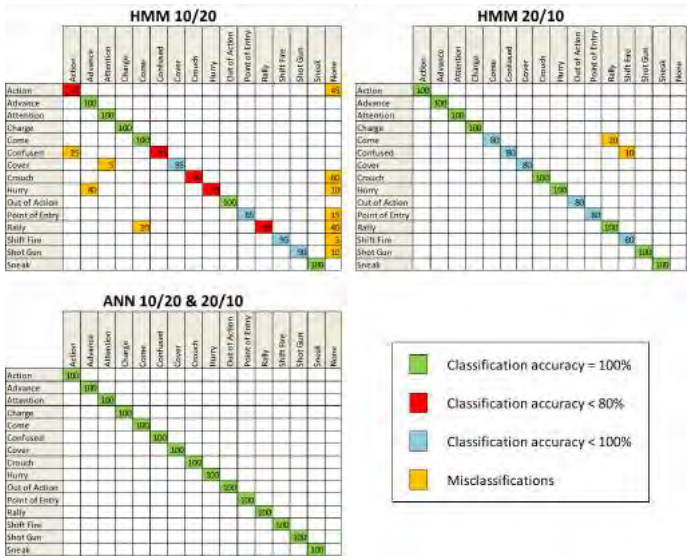



**Figure 6.7.** Confusion matrices for the CMU Dataset.

Across all gestures the ANN performs markedly better, achieving an accuracy of 100% for all gestures. Even with a smaller dataset, the performance of the ANN does not degrade. On the other hand, the HMM has an overall accuracy of 93.3% for the 20/10 test and an accuracy of 81.6% with the smaller training set.

Using HMM, there is confusion between the *Rally* and *Come* gestures. This result is seen in both the 20/10 test and the 10/20 test. With few training samples many of the gestures are misclassified by the 10/20 HMM. Five of the gestures have a correct classification rate of less than 80%. However, when doubling the number of training samples, all the gestures have a correct classification rate (CCR) of over this value.

This trend implies that as the amount of training data is increased, the HMM is able to build a better model to represent each gesture. The ANN can develop a model with fewer data samples. This result may be due to the optimised implementation

of the ANN, whereas the standard unoptimised algorithms were used for HMM training and testing.

To test the performance of the ANN the VisApp dataset was used. This dataset is presented in Section 3.1. The authors of the dataset use DTW for gesture classification. They divided the dataset into 8 samples performed by trained users and 20 samples performed by untrained users. The trained user data is use to train the DTW classifier and the untrained user data is used for system evaluation. In order to compare the results the same procedure is followed. The results can be seen in Figure 6.8.

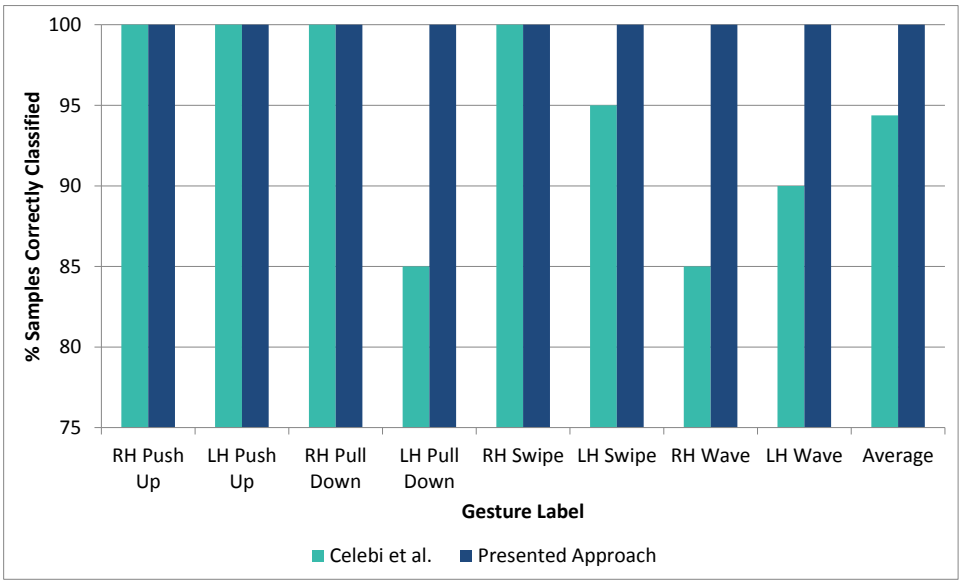



**Figure 6.8.** Comparison of the results obtained by Celebi et al. [28] and those obtained using the presented approach.

This dataset was also used to investigate the effect of cascading neural networks as presented in Figure 6.2. Figure 6.9 depicts the results of distinguishing between gesturing sides and the results if no distinction is made.

The confusion matrix for the recorded dataset can be seen in Figure 6.10.

The use of ANN results in a marked improvement in the results when compared to those obtained by Celebi et al. [28], particularly for the *Wave* gesture. There is also a 15% improvement from using an initial neural network to determine which side of the body is being used for gesture performance. A 100% classification accuracy is obtained across three different datasets. This result shows that the designed ANN is robust and can be used to classify dynamic upper body gestures.

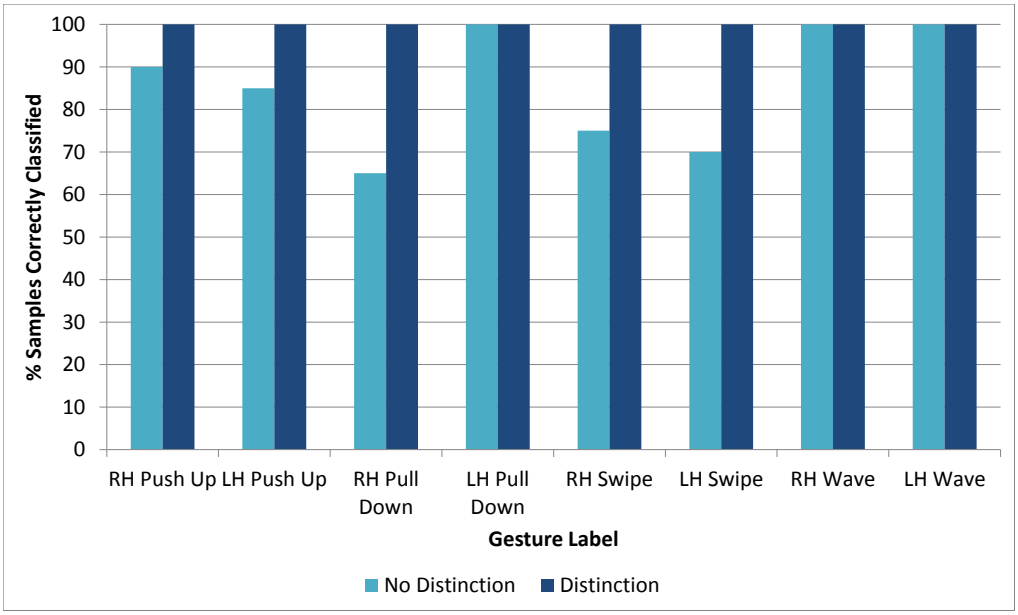



**Figure 6.9.** Comparison of the results obtained if no distinction between the left and right is made and if a distinction is made in the VisApp dataset.

## 6.2. Hand Gesture Recognition

The goal of the hand gesture recognition is to classify a hand gesture using the features extracted from an image, as described in Section 4.2. This section introduces two techniques used for hand gesture recognition, $k$-means and neural networks. HMMs are not employed as the hand gestures are static. Therefore, a temporal model is not required.

### 6.2.1. $K$-means Classification

$K$-means clustering is a method of identifying groups or clusters of data points in a multidimensional space. A cluster is a group of data points whose inter-point distance is small compared to the distances to points outside the cluster. $K$-means is used as a method to model the distribution of the data in a multidimensional space.

Given a dataset $\{x_1, ..., x_N\}$ consisting of $N$ observations of a $D$-dimensional feature vector $\mathbf{x}$. The goal of $k$-means is to assign each observation/data point one of K labels, such that these labels partition this dataset into $K$ clusters. A set of vectors $\mu_k$ are defined where $k = 1, ..., K$ in which $\mu_k$ represents the centre point of the $k^{\text{th}}$ cluster. $K$-means finds the set of vectors $\mu_k$ and the assignment of data points to clusters such that the sum of the squares of the distances of each data point to its closest vector $\mu_k$ is a minimum.

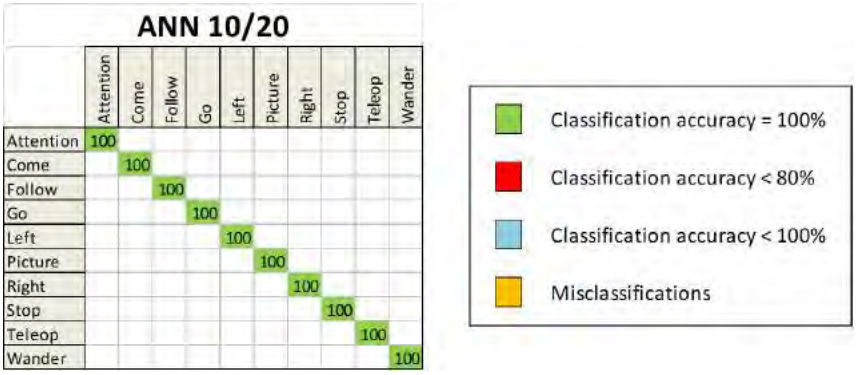



**Figure 6.10.** Confusion matrix for the CSIR Gesture Dataset using the 10/20 ANN.

A $N \times K$ matrix $R$ is defined that defines the assignment of data points to clusters. The element $r_{nk}$ of the matrix is a binary variable describing the assignment of the data point $x_n$ to the $k^{\text{th}}$ cluster. Each data point can only be assigned to a single cluster.

An objective function is defined, which represents the sum of the squared distances of each data point to its assigned centre as

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2. \tag{6.15}$$

The values of $\mu_k$ and $r_{nk}$ are updated iteratively so as to minimise $J$. This algorithm is shown in Algorithm 6.2 [113]. This is also a version of the EM algorithm.

**Algorithm 6.2** $K$-means

**function** $K$-MEANS($\mathbf{X}, K$) **return** $\mu_k, r_{nk}$
  **inputs**: $\mathbf{X}$, points in the dataset
    $K$, the number of clusters
  Initialise $\mu_k$ using the $k$++ algorithm (discussed later), shown in 6.3.
  **while** not converged **do**
    $r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j ||x_n - \mu_j||^2 \\ 0 & \text{otherwise} \end{cases}$
    $\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$
  **end while**
**end function**

The initial values of $\mu_k$ are found using the $k$++ algorithm [114] shown in Algorithm 6.3. This results in faster convergence as the points are not randomly assigned. Rather, $\mu_1$ is defined to be a randomly selected point from the dataset. Next the point with the minimum probability of belonging to the same cluster as this

point is found. This point is then assigned as the next point in $\mu_k$. This procedure is repeated until all the centres have been initialised.

**Algorithm 6.3** $k$++ Centre Initialisation

1: **function** $k$++ INITIALISATION($\mathbf{X}, K$) **return** $\mu_k$
2: **inputs**: $\mathbf{X}$, points in the dataset
3: $K$, the number of clusters
4: Take one centre $\mu_1$, chosen uniformly at random from $\mathbf{X}$.
5: **while** $i \neq K$ **do**
6: Calculate $D(x)$, the shortest distance from a data point to the closest centre already chosen.
7: Take a new centre $\mu_i$, choosing $x \in \mathbf{X}$ with probability $\frac{D(x)^2}{\sum_{x \in \mathbf{X}} D(x)^2}$.
8: **end while**
9: **end function**

**Determining the Number of Clusters** One of the pitfalls of the $k$-means algorithm is that the number of clusters must be specified *a-priori*. A number of methods exist, these are examined in the work by Milligan and Cooper [115]. The metric used for determining the number of clusters is the sum of squared error (SSE) metric. The SSE metric is chosen due to its simplicity. In this technique, the sum of squared error is calculated for each cluster and is plotted as a function of the number of clusters. The number of clusters is then chosen where adding another cluster does not give a much better result, often where an "elbow" occurs. The method is illustrated in Figure 6.11. A sample dataset was generated where the data has been sampled from three Gaussian mixture models. The cluster centres are shown in red and the circle represents the GMM with $\sigma = 1$, where $\sigma$ is the standard deviation. The number of clusters is at the value of $K$ where the is a "elbow" in the graph. In this case at $k = 3$.


**(a)** A sample dataset. **(b)** The SSE as a function of $K$.

**Figure 6.11.** Illustration of determining the number of clusters.

**K-means for Gesture Recognition** $K$-means is used to model the distribution of each static hand gesture in space. Rather than assuming that each gesture can be represented by a single cluster, consider that the gesture may have a complex manifold in space that may be represented by multiple clusters. A $k$-means classifier is trained for each of the twelve hand gestures to obtain cluster centres in the form $\{(C_1^1, ..., C_1^K), ..., (C_{12}^1, ..., C_{12}^K)\}$, where $k$ is the number of clusters per gesture. A new sample is classified as belonging to the gesture with the minimum distance.$k$ is not the same for each gesture as the method described in Section 6.2.1 is employed for each gesture.

For each new query, $F^Q$, the euclidean distance to each cluster centre is computed as,

$$\varepsilon(F^Q, C_g^k) := \sqrt{\sum_{m=1}^{M} (f_m^Q - C_{gm}^k)^2}, \tag{6.16}$$

where $M$ is the number of features, $C_g^k$ is the centre of the $k^{\text{th}}$ cluster of gesture $g$. The classification of a new query is then given by,

$$\begin{aligned} \text{label}(F^Q) &= \text{label}(F^w) \\ w &= \arg \min_{k=1..K, g=1..12} \varepsilon(F^Q, C_g^k) \end{aligned} \tag{6.17}$$

## 6.2.2. Neural Network for Hand Gesture Recognition

To compare the performance of the $k$-means classifier a neural network is implemented for hand gesture recognition. The network is a fully-connected, 3-layer neural network with one hidden layer. The input layer has 21 neurons, the hidden layer 24 neurons and the output layer 10 neurons. The network is trained using MBSGD, discussed in Section 6.1.1. The advantage of the neural network approach is that the weights of each feature can be tuned. For example, the position of the hand could have more weight than say the position of the elbow. On the other hand, in the $k$-means algorithm all features are weighted equally. In addition, $k$-means clustering tends to find clusters which have comparable spatial extent i.e. the clusters will have similar shapes.

## 6.2.3. Results

As discussed in Section 6.2, artificial neural networks (ANN) and $k$-means are used for hand gesture classification. This section presents the results for the recorded hand dataset. An overview of the dataset can be found in Section 3.2. A total of 15 subjects performed each of the twelve gestures three times, giving a total of 540 samples. However, in cases where the calibration phase failed (i.e. five fingers were not identified) the subsequent gestures were not used for classifier evaluation. After data pruning, there are 75 samples per gesture. Individual leave-one-out

cross validation (*ILOOCV*) is used to evaluate the performance of the classification system. In *ILOOCV*, the data is partitioned into $n$ subsets of equal size, where $n$ is the number of subjects. The classifier is then trained $n$ times using each of the $n$ subsets as the test set each time and the remaining data for training [116]. Therefore, the data is divided into 15 subsets. Each time 70 samples were used to train the classifier and 5 samples of each gesture was used for testing.

Three feature vectors were defined and compared, namely,

- *Hand joint positions*: this feature vector contains the normalised position of each of the hand joints.
- *Finger states*: this feature vector contains the state of each finger namely, whether the finger is *Straight*, *Bent*, *Closed*, and for the thumb, *Opposed* or *Extended*.
- *Hand joint angles*: this feature vector contains the finger joint angles discussed in Section 4.2.4.

It can be seen in Figure 6.12 that the correct classification rate across all gestures is the highest for the joint angle feature vector. This result is expected as the joint angles are rotation and translation invariant and can work for a large variety of users. The overall average accuracy using a ANN classifier is approximately 75%. The accuracy for the gestures is $57 - 100\%$.

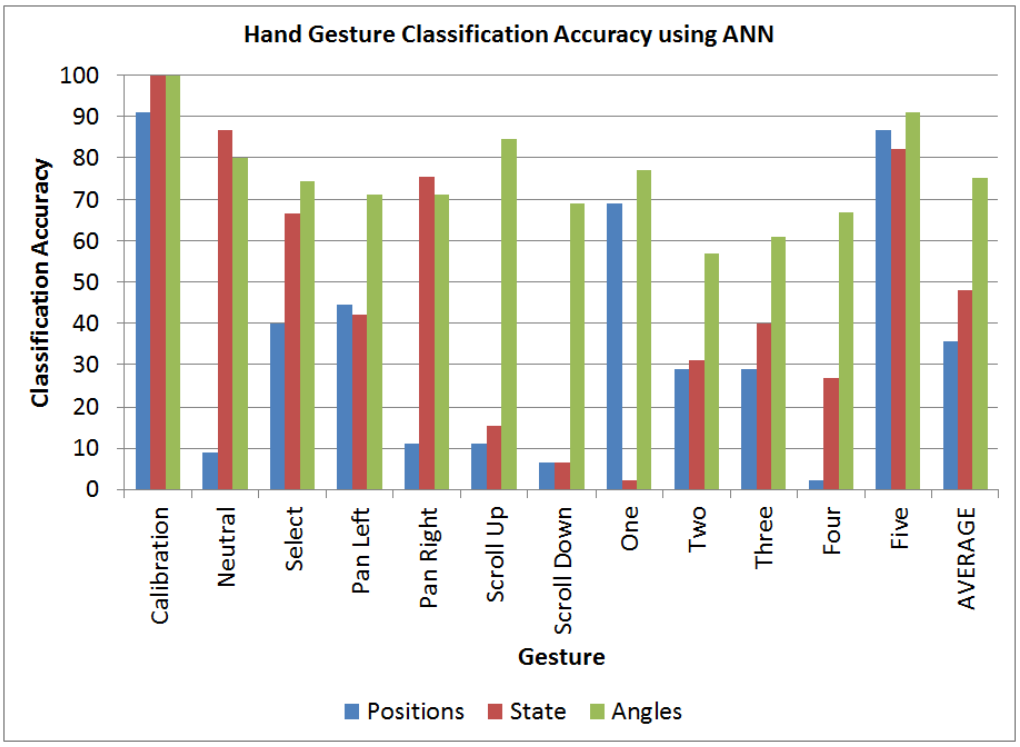



**Figure 6.12.** Correct classification rate using an ANN as a classifier.

Examining Figure 6.13, note that there is a large amount of confusion between the *Calibration* gesture and the *Five* gesture as they are identical. In calculating the correct classification rate gestures which belong to the *Calibration* class but are classified as belonging to gesture *Five* are considered as true positives. The same is true for gestures classified as *Calibration* but from the *Five* gesture class. Using the finger states feature vector the system is unable to distinguish between the *Scroll Up* gesture and the *Calibration* gesture. This is because the finger state parameters do not adequately describe all the possible configurations of the hand: the angle of abduction is not described using the hand joints. There is also a large degree of confusion between the *Neutral*, *Pan Left*, *Pan Right* and *Select* gestures. All of these gestures contain the extension of the thumb at the metacarpal-phalangeal joint, and in the case of the *Pan Right* and *Select* gestures the extension of one other finger. If this finger is not detected as being extended the gestures will have similar feature vectors.

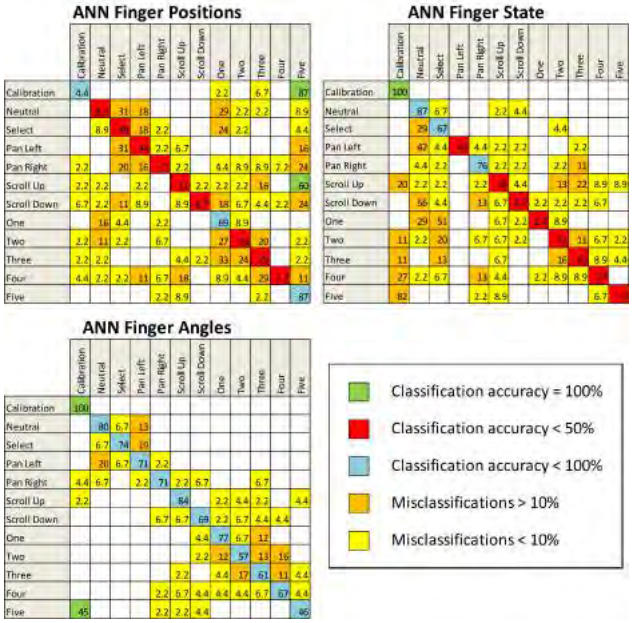



**Figure 6.13.** Confusion matrices when using an artificial neural network as a classifier for three different feature vectors.

Comparing the fingertip detection and the classification accuracy it can be seen that they are strongly correlated (see Figure 6.14). Each point on the graph represents a different gesture and the best fit line was determined using least square fitting. The line predicts 61.2% of the variance in the accuracy. Gestures with a lower fingertip detection have a lower classification accuracy, as expected.

A $k$-means classifier was trained for each of the twelve gestures. The sum of squared errors was used to determine the number of clusters for each gesture, as discussed in Section 6.2.1.

The same three feature vectors discussed above are used to compare the performance

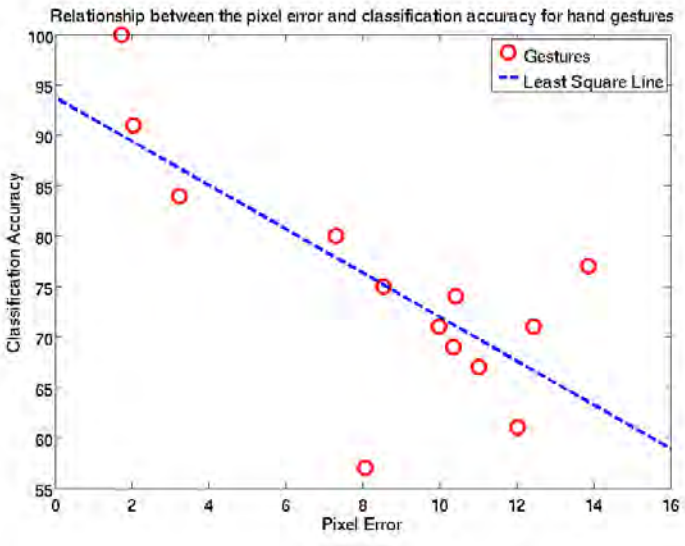



**Figure 6.14.** Relationship between the pixel error and the classification accuracy.

of the $k$-means classifier. The classification accuracy and confusion matrices are shown in Figures 6.15 and Figure 6.16 respectively. As for the ANN, the classification accuracy using the joint angle feature vector is significantly higher compared to the finger state and finger position feature vectors.

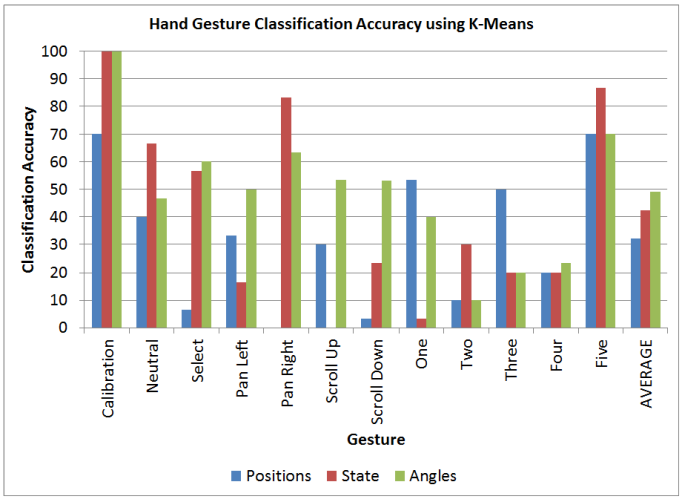



**Figure 6.15.** Comparison of the classification accuracy using a $K$-means classifier for three different feature vectors.

Figure 6.17 shows a comparison of the precision and recall for both the ANN and $k$-means classifiers using the joint angle feature vector. Note that both the precision and recall for the ANN is higher than for the $k$-means classifier. This is assumed to be due to the feature weighting employed by the ANN. The weighting for each feature is learnt during ANN training whilst for the $k$-means classifier each feature

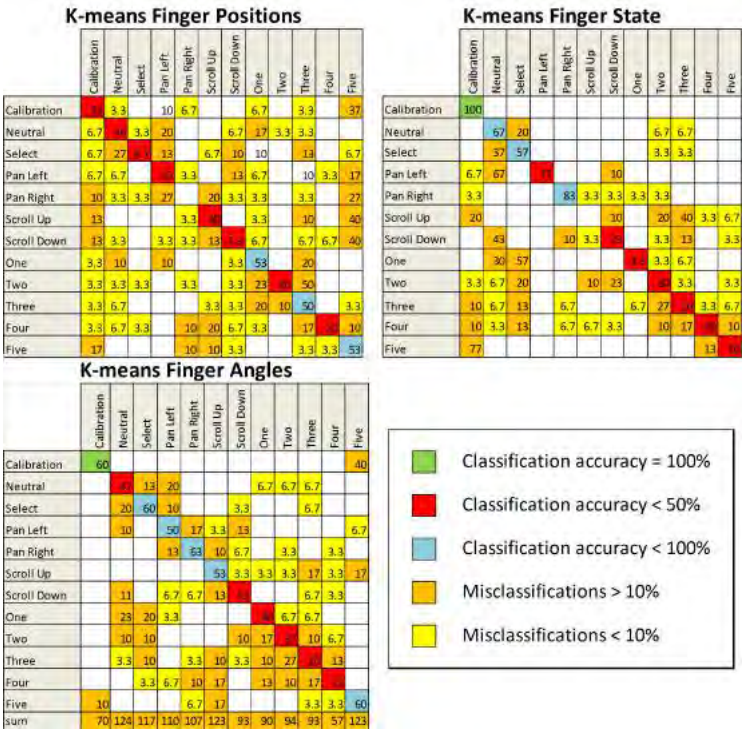



**Figure 6.16.** Confusion matrices for the $K$-means classifier.

is weighted equally.

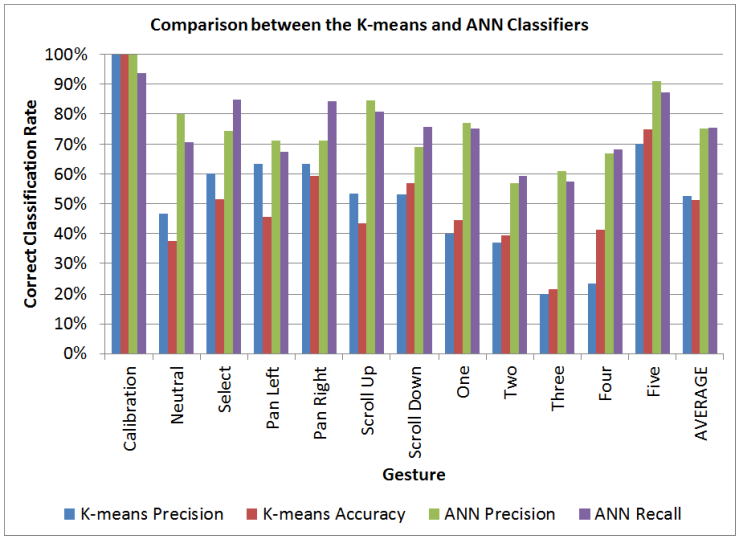



**Figure 6.17.** Comparison of the precision and recall for a $K$-means classifier and ANN.

## 6.3. Summary

This chapter discussed the classifiers used for gesture recognition. The neural networks and hidden Markov models used for upper body dynamic gesture recognition are presented. These are chosen to represent the temporal nature of dynamic gestures. The results show that cascading neural networks a classification accuracy of 100% is achieved across three different datasets. For the static hand gestures, $k$-means and neural networks are used. Unlike HMMs where the transition matrix is dependent on time, neural networks are not only used to model temporally dependent data, therefore the inputs to a neural network can be static. $K$-means is used to model the distribution of each gesture in multidimensional space, with the neural networks used as a comparative method. An analysis of the results reveals that the neural network classifier achieves better accuracy compared to the $k$-means classifier. This result is thought to be because the neural network tunes the weights of each feature, whereas the same weight is used for each feature in the $k$-means classifier. An improvement in the fingertip detection algorithms may increase the hand gesture recognition results as the pixel error is correlated to the classification accuracy for each gesture.

# 7. System Results

This chapter presents the implementation details of the system as well as the results for the integrated system. The implementation details are provided in Section 7.1 and results in Section 7.2.

## 7.1. System Implementation

All algorithms were implemented using C++. The specifications of the computer used can be seen in Table 7.1.

**Table 7.1.** Specifications of the computer used for implementing and testing the gesture recognition system.

| Property | Specification |
|---|---|
| Processor | Intel Core i7-3930 CPU @ 3.20 GHz |
| Memory | 16 GB |
| Operating System | Ubuntu 12.04 LTS |

OpenCV [117], the NiTE SDK [89] and OpenNI [105] were used to implement the vision algorithms presented in this work. The NiTE and OpenNI libraries are used to interface with the Kinect. The OpenNI library provides access to the depth and colour images and the NiTE library uses these images to calculate the user joint positions in real time. The OpenCV library provides a multitude of computer vision algorithms which can be used to further process the images. In particular the functionality provided by the core and highgui static libraries were used. These include functions such as findContours, and various drawing functions used for the GUI. The Eclipse IDE was used to create a project and integrate the libraries discussed above.

The Robot Operating System (ROS) Hydro [118] is used to control a Pioneer-3DX robot. Robot control experiments were simulated using MobileSim, a free simulator available from [119].

### 7.1.1. Viola-Jones Face Detector

Unlike hands, frontal views of faces have unique characteristics that make them easier to detect. The Viola-Jones [120] face detector uses a cascade of weak classifiers

to detect faces in an image. The detector is trained using multiple examples of faces. Rectangular filters similar to Haar wavelets are used to extract image features from an integral image. The OpenCV implementation was used to detect faces in the RGB image. If a frontal face is detected then it is assumed that the user is looking at the robot and gesture recognition is triggered. If multiple faces are detected, gesture recognition is triggered for the user closest to the robot.

### 7.1.2. Close-Far Boundary Detection

There are two modes of interacting with the robot, far-mode where upper body gestures are used to control the actions of the robot and near-mode where hand gestures are used to control a graphical interface attached to the robot. The mode activated depends on the distance of the user's hands from the camera. The distance boundaries and mode activations are depicted in Figure 7.1. At distances greater than 0.8 m far-mode interaction is activated. If the distance is less than 0.7 m, near-mode interaction is used instead. Between 0.7 to 0.8 m is a dead zone.There are dead zones where no mode is activated. Two of these dead zones are due to the range of the depth sensor, and the depth zone between the near-mode and the far-mode safeguards the system from switching continuously if the user is on the boundary between them.

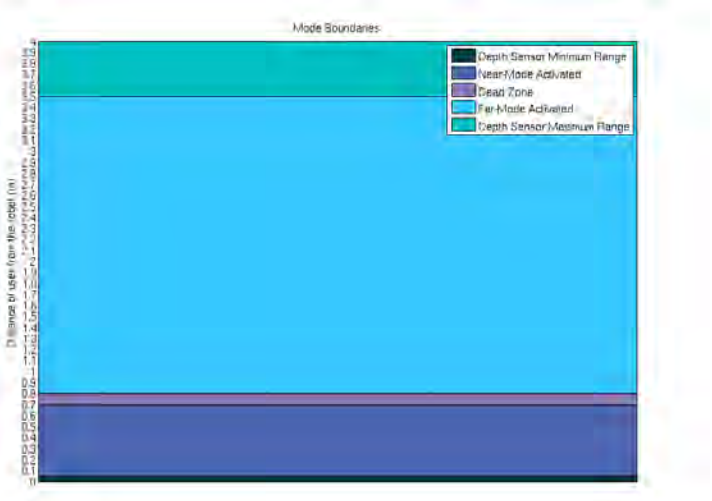



**Figure 7.1.** Distance boundaries for human-robot interaction.

## 7.2. Results

This section presents a summary of the results achieved for each component of the gesture recognition system.

### 7.2.1. Hand Gesture Recognition

The four components of the hand gesture recognition system are hand segmentation, fingertip detection, hand model generation or joint angle extraction and a classifier. This section provides a summary of the results obtained for each of these components.

The segmentation results show that a GCE less than 3% and LCE less than 2% is achieved by combing the colour, depth and hand orientation information. False positives, or background pixels classified as belonging to the hand, contribute the majority of the error. This error is due to the larger appearance of the hand in the depth image. Comparing this result to using only colour information for segmentation, a GCE of approximately 14% is achieved as there are many skin coloured objects in the background.

On average 88% of fingers are detected in the gesture images. The average for curvature detected fingers is 90.7% and the depth detection rate is 85.7%. These indicate that the detection methods achieve quite robust results. For gestures that contain primarily extended fingers, such as the *Calibration*, *Four* and *Five* gestures, the error is less than 5 pixels. A small error in this range is expected as the ground truth fingertips were marked manually. This result shows that the k-curvature detection algorithm is a robust and accurate method for detecting fingers that are extended and do not coincide with the internal contour of the hand. Gestures that contain tips in the hand contour have a larger error of approximately 9 pixels. Due the lower resolution of the depth sensor, internal contours in the hand are not as smooth as the external contour, which are generated using both depth and colour. This results in more tips being detected. Fingertips are then matched to the closest MCP position, and this could result in a larger error.

The estimated hand model was qualitatively compared to the pose of the hand, and the number of hand models that correctly matched the gesture were counted. On average, 80.8% of hand models were correctly generated using the fingertip positions and inverse kinematics.

The hand model is used to extract three feature vectors, namely the finger states, finger joint positions and the joint angles. The performance of a $k$-means classifier and ANN was compared for all three gesture. It is found that the joint angle feature vector achieves the highest classification accuracy for both classifiers. This is expected as the joint angles are rotation and translation invariant and can work for a large variety of users. Across all gestures, the ANN performance was greater than the classification performance of the $k$-means classifier. This result is assumed to be due to the feature weighting employed by the ANN. The weighting for each feature is learnt during ANN training while for the $k$-means classifier each feature is weighted equally. The overall average accuracy using an ANN classifier is approximately 75%. The accuracy for the gestures is $57 - 100\%$. The classification accuracy of gestures with extended fingers was higher than that where the fingers are flexed

or closed. This result is expected as the curvature-based fingertip detection has a smaller error compared to the depth-based fingertip detection. To achieve higher classification accuracy a higher resolution depth sensor must be used, or the depth-based detection method must be improved.

The live testing of the hand gesture recognition system can be seen in videos found in Appendix D. These videos show the implementation of a Rock, Paper, Scissors game. The objective of the game is to perform the gesture that either wins, loses or ties against the object shown on the screen and the instruction given. First the user is asked to raise their hand for calibration as shown in Figure 7.2.


**Figure 7.2.** Calibration GUI of the Rock, Paper, Scissors game

After calibration the game automatically starts. The GUI of the game and a description of each of the components can be seen in Figure 7.3. Of the 20 gestures in the demo, two gestures are misclassified. These can be seen in Figure 7.4. The hand gesture recognition system was able to operate in real-time.

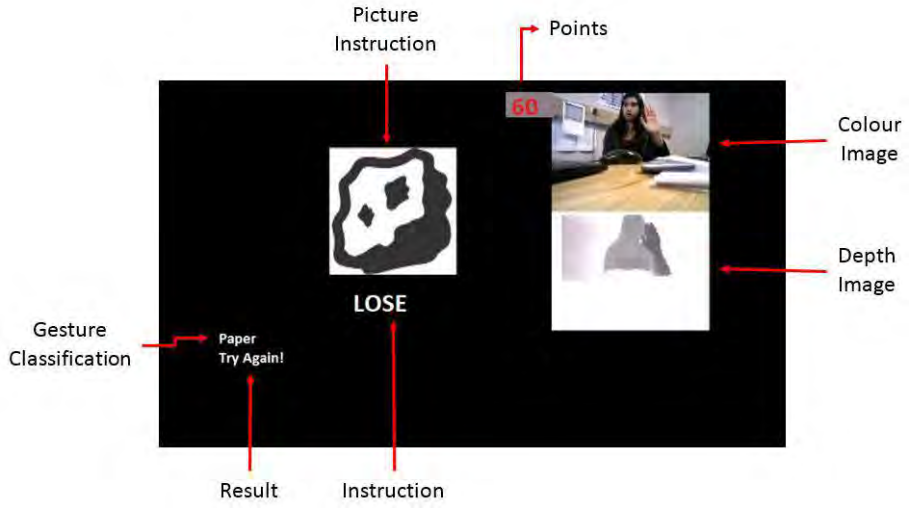
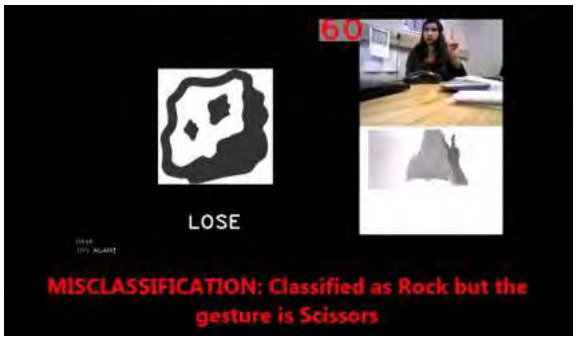
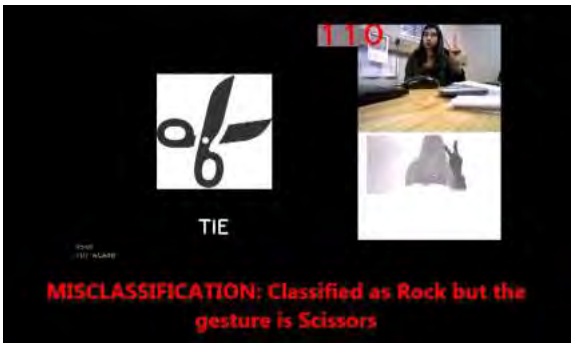



**Figure 7.3.** Description of the GUI elements for the Rock, Paper, Scissors game.




**(a)** Classified as rock but the gesture is scissors.




**(b)** Classified as rock but the gesture is scissors.

**Figure 7.4.** Misclassification's in the Rock, Paper, Scissors demo video.

### 7.2.2. Body Gesture Recognition

There are three components of the body gesture recognition system — feature extraction from the NiTE skeleton, temporal alignment of the gesture sequences and gesture classification. This section presents a summary of the results obtained for the alignment and classification components. The feature extraction was not evaluated as it is largely reliant on successful skeleton generation from the NiTE SDK and is assumed to be robust.

The results show that the key-frame selection method can reduce the log squared error by at least one order of magnitude after alignment. Qualitative results indicate that the method works for temporally aligning gesture sequences.

Two classifiers are compared for upper body gesture recognition — an ANN and HMM. The results show that cascading neural networks a classification accuracy of 100% is achieved across three different datasets. On the other hand, the HMM has an overall accuracy of 93.3% for the 20/10 test and an accuracy of 81.6% with the smaller training set. Increasing the amount of training data, the HMM can build a better model to represent each gesture. The ANN can develop a model with fewer data samples due to the optimised implementation of the ANN whereas the standard unoptimised algorithms were used for HMM training and testing. The use of ANN results in a marked improvement in the results when compared to those obtained by [28], particularly for the *Wave* gesture. There is also a 15% improvement from using an initial neural network to determine which side of the body is being used for gesture performance. These results show that the designed ANN is robust and can be used to classify dynamic upper body gestures.

The video of the live testing of the body gesture recognition system can be found in Appendix D. The video shows the actions of a simulated robot for different dynamic upper body gesture.

## 7.3. Summary

This chapter presented a summary of the results for the components of a gesture recognition system. The presented method for hand segmentation yields better results compared to using only depth or colour information and works for a wide variety of users even with a non-uniform background and with skin coloured objects in the background. This method satisfies user adaptability, reconfigurability and environmental requirements specified in Section 1.2. An accuracy of 75% was achieved using an ANN classifier with joint angles for hand-gesture recognition. This was shown to be an improvement over using the position of the fingertips or using a $k$-means classifier. The per-subject accuracy for the hand gesture recognition system has a small standard deviation, proving the robustness of the system for a variety of users. While the joint angle feature vector is superior to the position or state-based feature vectors this method relies on the correct detection of fingertips that are both fully extended. Further improvements to the fingertip detection algorithms must be made to improve the accuracy of the hand-gesture recognition system.The designed cascaded neural network architecture with MBSGD results in upper body gesture recognition of 100% across all gestures in three different datasets implying that the combination of ANNs with joint angles as a feature vector are ideal for use in classifying dynamic upper body gestures. Real-time performance is achieved for both the hand gestures and upper body gestures, fulfilling the responsiveness requirement specified in Section 1.2.

# 8. Conclusion and Future Work

## 8.1. Conclusion

This dissertation presents the design of a gestural interface for application in human-robot interaction. The design fuses upper body gestures for far-mode interaction, and hand poses for near-mode interaction depending on the distance of the user from the camera.

Far-mode interaction focuses on recognising upper body gestures using features extracted from skeletal data. A feature vector consisting of the joint angles of the upper body and the relative position of the hand and elbow joint is formed. A novel key-frame selection algorithm is proposed to temporally align the data. This approach is shown to decrease the intra-class log sum of squared error by at least one order of magnitude on both a recorded dataset and a publicly available dataset. In addition, a neural network is used to spot which side of the body is used for gesture performance, and to only extract features from the gesturing hand resulting in an improvement from 85% to 100%. Using a cascade of neural networks, the designed upper body gesture recognition system can recognise 100% of gestures in real-time for three different datasets.

Near-mode interaction focuses on retrieving the hand model using a single depth and colour image. The hand is segmented from the image using a colour model developed from the initial estimation of the hand position and depth segmentation. This method is shown to improve the segmentation results with a small global consistency error of less than 3% and a local consistency error of less than 2%. Fingers are detected in the image using the $k$-curvature algorithm, depth differences and parallel borders. A fingertip detection accuracy of 88% is achieved, but the pixel error for the depth-based segmentation is up to 15 pixels due to the low resolution of the depth camera. A novel calibration method insures that the hand model was invariant to changes in the anthropometric parameters of different users. Inverse kinematics are used to calculate the joint angles and hand model given the position of the fingertips and wrist points. The hand model matches the gesture in cases where the fingertips are correctly identified. The hand models can be calculated in real-time, fulfilling the responsiveness requirement of successful gesture recognition systems. A $k$-means classifier and neural network are used to classify gestures using the calculated joint angles. The neural network achieves a better accuracy than the $k$-means classifier.

The major contributions of this dissertation are:

1. A complete gestural interface which combines upper body gestures and hand gestures, and can therefore be used both when the user is far from the sensor and close to the sensor.
2. A novel key-frame selection algorithm for temporally aligning gesture sequences, reducing the $\log SSE$ error by one order of magnitude.
3. A complete upper body gestural interface, which combines a unique feature vector extracted from skeletal data and two neural networks, one for gesture spotting and another for gesture classification. Real-time, accurate, and robust upper body gesture recognition is achieved.
4. A novel hand model calibration method ensuring that the generated hand model matches the anthropometric measurements of the user.
5. Hand pose estimation implemented in real-time, combining fingertip detection and inverse kinematics. The use of joint angles allows for the gesture lexicon to be easily extended.

## 8.2. Future Work

For future work there are many possible improvements that could extend this work.

The current accuracy of the fingertip detection methods is approximately 88%. An improvement will result in a higher hand pose estimation accuracy. A possible optimisation includes using the latest generation of depth sensors to improve the depth resolution available.

Currently, hand pose estimation only works when the hand is orientated with the palm approximately parallel to the sensor. The hand pose estimation could be extended to include instances where the hand is perpendicular to the sensor. This extension could increase the number of possible gestures. However, this is a challenging task as the fingers may be occluded by one another.

One of the primary applications of hand gesture recognition is sign language translation. Sign languages typically include both dynamic and static gestures, so adapting the system to recognise dynamic gestures would increase the applicability of the system.

# Bibliography


[1] E. Guizzo. So, where are my robot servants? *Spectrum, IEEE*, 51(6):74–79, June 2014.

[2] Evan Ackerman. OSHbot will save you from having to ask for help in a hardware store. Available at `http://spectrum.ieee.org/automaton/robotics/industrial-robots/oshbot-will-save-you-from-asking-for-help-in-a-hardware-store`.

[3] Richard A. Bolt. *"Put-that-there": Voice and gesture at the graphics interface*, volume 14. ACM, 1980.

[4] A. Kendon. *Gesture: visible action as utterance*. Cambridge University Press, Cambridge, 2004.

[5] Daniel Casasanto. Gesture and Language Processing. In H. Pashler, T. Crane, M. Kinsbourne, F. Ferreira, and R. Zemel, editors, *Encyclopedia of the Mind*, pages 372–374. Oxford University Press, New York, 2013.

[6] Juan Pablo Wachs, Mathias Kölsch, Helman Stern, and Yael Edan. Vision-based hand-gesture applications. *Communications of the ACM*, 54(2):60, February 2011.

[7] D. Vishnu Vardhan and P. Penchala Prasad. Hand gesture recognition application for physically disabled people. *International Journal of Science and Research*, 3(8):765–769, 2014.

[8] Prapat Parab, Sanika Kinalekar, Rohit Chavan, Deep Sharan, and Shubhadha Deshpande. Hand gesture recognition using microcontroller and flex sensor. *International Journal Of Scientific Research And Education*, 2(03), 2014.

[9] Farid Parvini, Dennis McLeod, Cyrus Shahabi, Bahareh Navai, Baharak Zali, and Shahram Ghandeharizadeh. An approach to glove-based gesture recognition. In *Human-Computer Interaction. Novel Interaction Methods and Techniques*, pages 236–245. Springer, 2009.

[10] Thomas Allevard, Eric Benoit, and Laurent Foulloy. Fuzzy glove for gesture recognition. In *XVII IMEKO World Congress*, pages 2026–2031, 2003.

[11] Manisha R. Ghunawat. Multi-point gesture recognition using LED gloves for interactive HCI. *International Journal of Computer Science & Information Technologies*, 5(5), 2014.

[12] Mario Ganzeboom. How hand gestures are recognized using a dataglove. *Human Media Interaction (HMI)*, 2009.

[13] D. Vishnu Vardhan and P. Penchala Prasad. Hand gesture recognition application for physically disabled people. *International Journal of Science and Research*, 3(8):765–769, 2014.

[14] Amit Gupta, Vijay Kumar Sehrawat, and Mamta Khosla. FPGA based real time human hand gesture recognition system. *Procedia Technology*, 6:98–107, 2012.

[15] Mohammed Hasanuzzaman, Vuthichai Ampornaramveth, Tao Zhang, M.A. Bhuiyan, Yoshiaki Shirai, and Haruki Ueno. Real-time vision-based gesture recognition for human robot interaction. In *IEEE International Conference on Robotics and Biomimetics, 2004. ROBIO 2004.*, pages 413–418. IEEE, 2004.

[16] N. Naidoo and James Connan. Gesture recognition using feature vectors. In *Proc. South African Telecommunication Networks and Applications Conference (SATNAC 2009)*, 2009.

[17] Thad Starner, Joshua Weaver, and Alex Pentland. Real-time American sign language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371–1375, 1998.

[18] Tasnuva Ahmed. A neural network based real time hand gesture recognition system. *International Journal of Computer Applications*, 59(4):17–22, 2012.

[19] William T. Freeman and Michal Roth. Orientation histograms for hand gesture recognition. In *International Workshop on Automatic Face and Gesture Recognition*, volume 12, pages 296–301, 1995.

[20] Juan P. Wachs, Helman Stern, and Yael Edan. Cluster labeling and parameter estimation for the automated setup of a hand-gesture recognition system. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 35(6):932–944, 2005.

[21] Raymond Lockton and Andrew W. Fitzgibbon. Real-time gesture recognition using deterministic boosting. In *BMVC*, volume 2002, pages 1–10, 2002.

[22] Alex Drake. Kinect hand recognition and tracking. *Washington University in St. Louis, Kinect Hand Recognition and Tracking, Project Report*, 2012.

[23] James Davis and Mubarak Shah. Visual gesture recognition. In *Vision, Image and Signal Processing, IEE Proceedings*, volume 141, pages 101–106. IET, 1994.

[24] Zhou Ren, Jingjing Meng, and Junsong Yuan. Depth camera based hand gesture recognition and its applications in human-computer-interaction. In *2011 8th International Conference on Information, Communications and Signal Processing (ICICS)*, pages 1–5, Dec 2011.

[25] J.L. Raheja, A. Chaudhary, and K. Singal. Tracking of fingertips and centers of palm using Kinect. In *2011 Third International Conference on Computational Intelligence, Modelling and Simulation (CIMSiM)*, pages 248–252, Sept 2011.

[26] Yi Li. Hand gesture recognition using Kinect. In *2012 IEEE 3rd International Conference on Software Engineering and Service Science (ICSESS)*, pages 196–199. IEEE, 2012.

[27] Yuan Yao and Yun Fu. Contour model based hand-gesture recognition using Kinect sensor. *IEEE Transactions on Circuits and Systems for Video Technology*, 2014.

[28] Sait Celebi, Ali S.A.S. Aydin, T.T. Talha T. Temiz, and Tarik Arici. Gesture recognition using skeleton data with weighted dynamic time warping. *Computer Vision Theory and and Applications. VisApp*, 2013.

[29] Lee Jaemin, Hironori Takimoto, Hitoshi Yamauchi, A Kanazawa, and Yasue Mitsukura. A robust gesture recognition based on depth data. In *2013 19th Korea-Japan Joint Workshop on Frontiers of Computer Vision,(FCV)*, pages 127–132. IEEE, 2013.

[30] Garrett Bernstein, Nyk Lotocky, and Dan Gallagher. Robot Recognition of Military Gestures CS 4758 Term Project. Technical report, Robot Learning Lab, Cornell University, 2012.

[31] Zahoor Zafrulla, Helene Brashear, Thad Starner, Harley Hamilton, and Peter Presti. American sign language recognition with the Kinect. In *Proceedings of the 13th international conference on multimodal interfaces*, pages 279–286. ACM, 2011.

[32] Kam Lai, Janusz Konrad, and Prakash Ishwar. A gesture-driven computer interface using Kinect. In *2012 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI)*, pages 185–188. IEEE, 2012.

[33] J. Suarez and R.R. Murphy. Hand gesture recognition with depth images: A review. In *RO-MAN, 2012 IEEE*, pages 411–417, Sept 2012.

[34] Barry Kollee, Sven Kratz, and Anthony Dunnigan. Exploring gestural interaction in smart spaces using head mounted devices with ego-centric sensing. In *Proceedings of the 2nd ACM Symposium on Spatial User Interaction*, SUI'14, pages 40–49, New York, NY, USA, 2014. ACM.

[35] Sven Kratz and M.D.T. I. Aumi. AirAuth: A Biometric Authentication System Using In-air Hand Gestures. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '14, pages 499–502, New York, NY, USA, 2014. ACM.

[36] Tomás Mantecón, Carlos R. del Blanco, Fernando Jaureguizar, and Narciso García. New generation of human machine interfaces for controlling UAV through depth-based gesture recognition. In *SPIE Defense+ Security*, pages 90840C–90840C. International Society for Optics and Photonics, 2014.

[37] Ondrej Kainz and František Jakab. Approach to hand tracking and gesture recognition based on depth-sensing cameras and EMG monitoring. *Acta Informatica Pragensia*, 3(1):104–112, 2014.

[38] Frank Weichert, Daniel Bachmann, Bartholomäus Rudak, and Denis Fisseler. Analysis of the accuracy and robustness of the Leap motion controller. *Sensors*, 13(5):6380–6393, 2013.

[39] Maryam Khademi, Hossein Mousavi Hondori, Alison McKenzie, Lucy Dodakian, Cristina Videira Lopes, and Steven C. Cramer. Free-hand interaction with Leap motion controller for stroke rehabilitation. In *CHI'14 Extended Abstracts on Human Factors in Computing Systems*, pages 1663–1668. ACM, 2014.

[40] Ali Boyali, Naohisa Hashimoto, and Osamu Matsumato. Hand posture control of a robotic wheelchair using a Leap motion sensor and block sparse representation based classification. In *SMART 2014, The Third International Conference on Smart Systems, Devices and Technologies*, pages 20–25, 2014.

[41] D. Bassily, C. Georgoulas, J. Guettler, T. Linner, and T. Bock. Intuitive and adaptive robotic arm manipulation using the Leap motion controller. In *Proceedings of ISR/Robotik 2014; 41st International Symposium on Robotics*, pages 1–7. VDE, 2014.

[42] Zhou Ren, Jingjing Meng, Junsong Yuan, and Zhengyou Zhang. Robust hand gesture recognition with Kinect sensor. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 759–760. ACM, 2011.

[43] Nicolas Pugeault and Richard Bowden. Spelling it out: Real-time ASL fingerspelling recognition. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1114–1119. IEEE, 2011.

[44] Aditya Ramamoorthy, Namrata Vaswani, Santanu Chaudhury, and Subhashis Banerjee. Recognition of dynamic hand gestures. *Pattern Recognition*, 36(9):2069–2081, 2003.

[45] Matthew Tang. Recognizing hand gestures with Microsoft's Kinect. *Palo Alto: Department of Electrical Engineering of Stanford University*, 2011.

[46] Chieh-Chih Wang and Ko-Chih Wang. Hand posture recognition using Adaboost with SIFT for human robot interaction. In *Recent progress in robotics: viable robotic service to human*, pages 317–329. Springer, 2008.

[47] Pallavi Gurjal and Kiran Kunnur. Real time hand gesture recognition using SIFT. *International Journal for Electronics and Engineering*, pages 19–33, 2012.

[48] Qing Chen, Nicolas D. Georganas, and Emil M. Petriu. Real-time vision-based hand gesture recognition using Haar-like features. In *Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007. IEEE*, pages 1–6. IEEE, 2007.

[49] Qing Chen, Nicolas D. Georganas, and Emil M. Petriu. Hand gesture recognition using Haar-like features and a stochastic context-free grammar. *IEEE Transactions on Instrumentation and Measurement*, 57(8):1562–1571, 2008.

[50] Chen-Chiung Hsieh and Dung-Hua Liou. Novel Haar features for real-time hand gesture recognition using SVM. *Journal of Real-Time Image Processing*, pages 1–14, 2012.

[51] Kai-ping Feng and Fang Yuan. Static hand gesture recognition based on HOG characters and support vector machines. In *2013 2nd International Symposium on Instrumentation and Measurement, Sensor Network and Automation (IMSNA)*, pages 936–938. IEEE, 2013.

[52] Hui Li, Lei Yang, Xiaoyu Wu, Shengmiao Xu, and Youwen Wang. Static hand gesture recognition based on HOG with Kinect. In *2012 4th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, volume 1, pages 271–273. IEEE, 2012.

[53] Yu Ren and Chengcheng G.U. Hand gesture recognition based on HOG characters and SVM. *Bulletin of Science and Technology*, 2:011, 2011.

[54] Ankit Chaudhary, Jagdish L. Raheja, and Shekhar Raheja. A vision based geometrical method to find fingers positions in real time hand gesture recognition. *Journal of Software*, 7(4):861–869, 2012.

[55] Guan-Feng He, Sun-Kyung Kang, Won-Chang Song, and Sung-Tae Jung. Real-time gesture recognition using 3D depth camera. In *2011 IEEE 2nd International Conference on Software Engineering and Service Science (ICSESS)*, pages 187–190. IEEE, 2011.

[56] Unseok Lee and Jiro Tanaka. Finger identification and hand gesture recognition techniques for natural user interface. In *Proceedings of the 11th Asia Pacific Conference on Computer Human Interaction*, pages 274–279. ACM, 2013.

[57] Zhou Ren, Junsong Yuan, Jingjing Meng, and Zhengyou Zhang. Robust part-based hand gesture recognition using Kinect sensor. *IEEE Transactions on Multimedia*, 15(5):1110–1120, 2013.

[58] Shuxin Qin, Xiaoyang Zhu, Yiping Yang, and Yongshi Jiang. Real-time hand gesture recognition from depth images using convex shape decomposition method. *Journal of Signal Processing Systems*, 74(1):47–58, 2014.

[59] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A. Argyros. Efficient model-based 3D tracking of hand articulations using Kinect. In *BMVC*, volume 1, page 3, 2011.

[60] James Anthony Brown and David W. Capson. A framework for 3D model-based visual tracking using a GPU-accelerated particle filter. *IEEE Transactions on Visualization and Computer Graphics*, 18(1):68–80, 2012.

[61] Vassilis Athitsos and Stan Sclaroff. Estimating 3D hand pose from a cluttered image. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–432. IEEE, 2003.

[62] Paul Doliotis, Vassilis Athitsos, Dimitrios Kosmopoulos, and Stavros Perantonis. Hand shape and 3D pose estimation using depth data from a single cluttered frame. In *Advances in Visual Computing*, pages 148–158. Springer, 2012.

[63] B. Stenger, P.R.S. Mendonca, and R. Cipolla. Model-based 3D tracking of an articulated hand. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001. CVPR 2001.*, volume 2, pages II–310–II–315 vol.2, 2001.

[64] Javier Molina and José M. Martínez. A synthetic training framework for providing gesture scalability to 2.5 D pose-based hand gesture recognition systems. *Machine Vision and Applications*, pages 1–7, 2014.

[65] Stefan Waldherr, Roseli Romero, and Sebastian Thrun. A gesture based interface for human-robot interaction. *Autonomous Robots*, 9(2):151–173, 2000.

[66] Cem Keskin, Furkan Kıraç, Yunus Emre Kara, and Lale Akarun. Real time hand pose estimation using depth sensors. In *Consumer Depth Cameras for Computer Vision*, pages 119–137. Springer, 2013.

[67] Chen Yiqiang, G.A.O Wen, and M.A. Jiyong. Hand gesture recognition based on decision tree.

[68] Xiaoming Yin and Ming Xie. Finger identification and hand posture recognition for human–robot interaction. *Image and Vision Computing*, 25(8):1291–1300, 2007.

[69] Xinshuang Zhao, Ahmed M. Naguib, and Sukhan Lee. Kinect based calling gesture recognition for taking order service of elderly care robot. In *2014 RO-MAN: The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pages 525–530. IEEE, 2014.

[70] Chen Qian, Xiao Sun, Yichen Wei, Xiaoou Tang, and Jian Sun. Realtime and robust hand tracking from depth.

[71] Osamu Sugiyama, Takayuki Kanda, Michita Imai, Hiroshi Ishiguro, and Norihiro Hagita. Natural deictic communication with humanoid robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007. IROS 2007.*, pages 1441–1448. IEEE, 2007.

[72] Vijay Kumari Thakur. Robust hand gesture recognition for human machine interaction system. *Journal of Global Research in Computer Science*, 5(3):14–19, 2014.

[73] Paulo Trigueiros, António Fernando Ribeiro, and Gil Lopes. Vision-based hand segmentation techniques for human-robot interaction for real-time applications. 2012.

[74] M.D. Hasanuzzaman and Haruki Ueno. Face and gesture recognition for human-robot interaction. *Face Recognition*, page 149.

[75] Michael van den Bergh, Daniel Carton, Roderick de Nijs, Nikos Mitsou, Christian Landsiedel, Kolja Kuehnlenz, Dirk Wollherr, Luc van Gool, and Martin Buss. Real-time 3D hand gesture interaction with a robot for understanding directions from humans. In *RO-MAN, 2011 IEEE*, pages 357–362. IEEE, 2011.

[76] Dan Xu, Xinyu Wu, Yen-Lun Chen, and Yangsheng Xu. Online dynamic gesture recognition for human robot interaction. *Journal of Intelligent & Robotic Systems*, pages 1–14, 2014.

[77] Hee-Deok Yang, A-Yeon Park, and Seong-Whan Lee. Gesture spotting and recognition for human–robot interaction. *IEEE Transactions on Robotics*, 23(2):256–270, 2007.

[78] Dongseok Yang, Jong-Kuk Lim, and Younggeun Choi. Early childhood education by hand gesture recognition using a smartphone based robot. In *2014 RO-MAN: The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pages 987–992. IEEE, 2014.

[79] Pujan Ziaie. Implementing and Evaluating Hand Gesture Recognition for Human-Robot Joint Action. Master's thesis, Technische Universität München, Fakultät für Informatik, Munich, Germany, 2008.

[80] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.

[81] Jakub Segen and Senthil Kumar. Fast and accurate 3D gesture recognition interface. In *Fourteenth International Conference on Pattern Recognition, 1998. Proceedings.*, volume 1, pages 86–91. IEEE, 1998.

[82] Mohammed Yeasin and Subhasis Chaudhuri. Visual understanding of dynamic hand gestures. *Pattern Recognition*, 33(11):1805–1817, 2000.

[83] Alena Kopaničáková and Mária Virčíková. Gesture recognition using DTW and its application potential in human-centered robotics. *Robotics research paper*, 2013.

[84] Simon Fothergill, Helena Mentis, Pushmeet Kohli, and Sebastian Nowozin. Instructing people for training gestural interactive systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1737–1746. ACM, 2012.

[85] Zhe Lin, Zhuolin Jiang, and Larry S. Davis. Recognizing actions by shape-motion prototype trees. In *2009 IEEE 12th International Conference on Computer Vision*, pages 444–451. IEEE, 2009.

[86] Isabelle Guyon, Vassilis Athitsos, Pat Jangyodsuk, Ben Hamner, and Hugo Jair Escalante. Chalearn gesture challenge: Design and first results. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1–6. IEEE, 2012.

[87] Sergio Escalera, Xavier Baró, Jordi Gonzalez, Miguel A. Bautista, Meysam Madadi, Miguel Reyes, V. Ponce, Hugo J. Escalante, Jamie Shotton, and Isabelle Guyon. Chalearn looking at people challenge 2014: Dataset and results. In *ECCV ChaLearn Workshop on Looking at People*, 2014.

[88] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics*, 33, August 2014.

[89] PrimeSense. NiTE 2 API Programmer Tutorial Guide, 2013. Available from `http://www.primesense.com/wp-content/uploads/2013/04/PrimeSense_NiTE2API_ProgTutorialGuide_C++Samples_docver0.2.pdf`.

[90] Jure Kovac, Peter Peer, and Franc Solina. *Human skin color clustering for face detection*, volume 2. IEEE, 2003.

[91] Teófilo de Campos. *3D Visual Tracking of Articulated Objects and Hands*. PhD thesis, University of Oxford, Oxford, United Kingdom, 7 2006. Available from `http://www.robots.ox.ac.uk/~teo/thesis/`.

[92] Marko Tkalcic and Jurij F. Tasic. Colour spaces: perceptual, historical and applicational background. In *Eurocon*, 2003.

[93] Bosheng Wang and Jiaqi Xu. Accurate and fast hand-forearm segmentation algorithm based on silhouette. In *2012 IEEE 2nd International Conference on Cloud Computing and Intelligent Systems (CCIS)*, volume 2, pages 976–979. IEEE, 2012.

[94] Gunilla Borgefors. Distance transformations in digital images. *Computer vision, graphics, and image processing*, 34(3):344–371, 1986.

[95] Hui Liang, Junsong Yuan, and Daniel Thalmann. 3D fingertip and palm tracking in depth image sequences. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 785–788. ACM, 2012.

[96] Satoshi Suzuki. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.

[97] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings. Eighth IEEE International Conference on Computer Vision, 2001. ICCV 2001.*, volume 2, pages 416–423. IEEE, 2001.

[98] Zhenyao Mo and Ulrich Neumann. Real-time hand pose recognition using low-resolution depth images. In *CVPR (2)*, pages 1499–1505, 2006.

[99] Chin-Seng Chua, Haiying Guan, and Yeong-Khing Ho. Model-based 3D hand posture estimation from a single 2D image. *Image and Vision computing*, 20(3):191–202, 2002.

[100] James J. Kuch and Thomas S. Huang. Vision based hand modeling and tracking for virtual teleconferencing and telecollaboration. In *Fifth International Conference on Computer Vision, 1995. Proceedings.*, pages 666–671. IEEE, 1995.

[101] Jintae Lee and Tosiyasu L. Kunii. Constraint-based hand animation. In *Models and techniques in computer animation*, pages 110–127. Springer, 1993.

[102] Buryanov Alexander and Kotiuk Viktor. Proportions of hand segments. *Int. J. Morphol*, 28(3):755–758, 2010.

[103] Hans Rijpkema and Michael Girard. Computer animation of knowledge-based human grasping. In *ACM Siggraph Computer Graphics*, volume 25, pages 339–348. ACM, 1991.

[104] Asus. Asus Xtion PRO Live, 2012. Available from `http://www.asus.com/Multimedia/Xtion_PRO_LIVE/`.

[105] OpenNI. OpenNI Programmer's Guide, 2013. Available from `http://www.openni.org/openni-programmers-guide`.

[106] R. Mangera. Static gesture recognition using features extracted from skeletal data. In *Proceedings of the Twenty-Fourth Annual Symposium of the Pattern Recognition Association of South Africa(PRASA2013)*, pages 59–63. PRASA, 2013.

[107] Jun Cheng, Wei Bian, and Dacheng Tao. Locally regularized sliced inverse regression based 3D hand gesture recognition on a dance robot. *Information Sciences*, 221:274–283, 2013.

[108] Meinard Müller. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84, 2007.

[109] Chotirat Ann Ratanamahatana and Eamonn Keogh. Three myths about dynamic time warping data mining. In *Proceedings of SIAM International Conference on Data Mining (SDM'05)*, pages 506–510. SIAM, 2005.

[110] Stuart Jonathan Russell, Peter Norvig, John F. Canny, Jitendra M. Malik, and Douglas D. Edwards. *Artificial intelligence: a modern approach*, volume 74. Prentice hall Englewood Cliffs, 1995.

[111] I. Nabney. *NETLAB: Algorithms for Pattern Recognition*. Advances in Computer Vision and Pattern Recognition. Springer, 2002.

[112] Alexander Fabisch. OpenANN, 2013. Available from `http://openann.github.io/OpenANN-apidoc/index.html`.

[113] Christopher M. Bishop et al. *Pattern recognition and machine learning*, volume 4. springer New York, 2006.

[114] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium*

*on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[115] Glenn W. Milligan and Martha C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, 1985.

[116] Tom M. Mitchell. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45, 1997.

[117] OpenCV. OpenCV API Reference. Available from `http://docs.opencv.org/modules/refman.html`, year = 2014.

[118] Open Source Robotics Foundation. ROS Hydro Medusa, 2014. Available from `http://wiki.ros.org/hydro`.

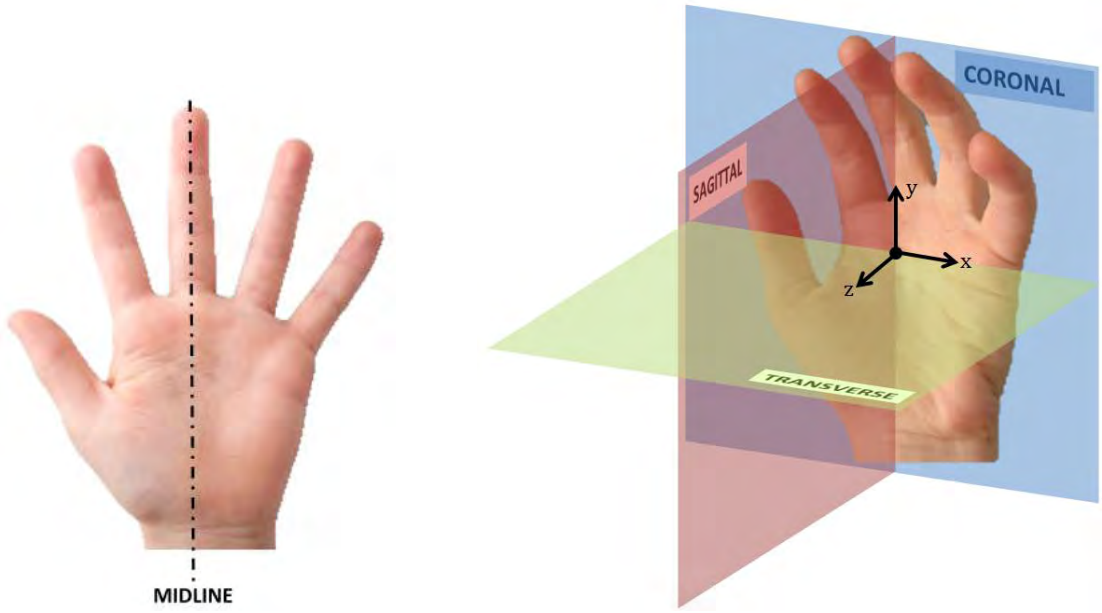
[119] Adept Technology. MobileSim— The Adept MobileRobots Simulator, 2012. Available from `http://robots.mobilerobots.com/MobileSim/download/current/README.html`.

[120] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001. CVPR 2001.*, volume 1, pages I–511. IEEE, 2001.

# A. Anatomical Terms of Motion

## A.1. Overview

Anatomical terms of motion are used to describe the movement of joints of the body relative to their anatomical position. As the majority of motions have an inverse, for example a push and pull, they are paired. This section describes the terms of motion, specifically those that are applicable to the hand. The planes and terms used to describe the direction of motion are shown in Figure A.1. The midline refers to the middle of the hand, as depicted in Figure A.1a. Three planes are commonly used, as shown in Figure A.1b. The sagittal plane is through the mid-line, dividing the palm into left and right. The coronal plane is parallel to the palm and divides the hand into front and back, and lastly the transverse plane divides the hand into top and bottom.



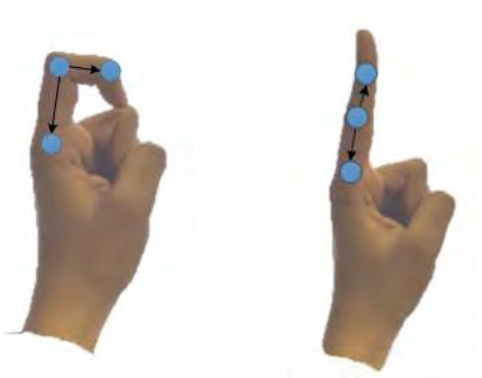

**(a)** The midline of the hand. **(b)** The three anatomical planes.

**Figure A.1.** The anatomical planes and terminology used to describe motion.

## A.2. Flexion-Extension

Flexion and extension are movements that occur in the sagittal plane. They are actions that bend or straighten a part of the body, thereby increasing or decreasing the angle between two subsequent segments. Flexion decreases the angle and extension increases the angle. These motions are depicted in Figure A.2.

Flexion and extension of the hand at the wrist joint is referred to as palmarflexion and dorsiflexion. Palmarflexion refers to decreasing the angle between the palm and anterior forearm and dorsiflexion is the extension of the wrist joint.



**(a)** Flexion of the proximal interphalangeal joint.

**(b)** Extension of the proximal interphalangeal joint.

**Figure A.2.** Flexion-Extension.

## A.3. Abduction-Adduction

Abduction and adduction describe the motion of a body part away from or towards the midline. In the case of the hand, the midline is the middle of the hand, as shown in Figure A.1a. Abduction is movement away from the midline, for example when the fingers are spread, and adduction is movement towards the midline. These motions are illustrated in Figure A.3.

## A.4. Pronation-Supination

Pronation is rotation of the forearm such that the palm faces towards the rear and supination is a rotation of the forearm so that the palm faces towards the anterior. These are shown in Figure A.4.

**(a)** Abduction of the index finger.

**(b)** Adduction of the index finger.

**Figure A.3.** Abduction-Adduction.

**(a)** Pronation.

**(b)** Supination.

**Figure A.4.** Pronation-Supination.

## A.5. Opposition-Reposition

These movements are unique to humans and are only applicable to the carpal-metacarpal joint of the thumb. Opposition brings the thumb and little finger together in a grasping motion and reposition is the opposite movement, which moves the thumb and little finger apart. An example of these motions is depicted in Figure A.5.

**Figure A.5.** Opposition of the thumb.

# B. Ethics Clearance

This chapter contains the ethics clearance certificates received from the University of Cape Town (UCT) and the Council for Scientific and Industrial Research (CSIR) for this work.

# EBE Faculty: Assessment of Ethics in Research Projects (Rev2)

Any person planning to undertake research in the Faculty of Engineering and the Built Environment at the University of Cape Town is required to complete this form before collecting or analysing data. When completed it should be submitted to the supervisor (where applicable) and from there to the Head of Department. If any of the questions below have been answered YES, and the applicant is NOT a fourth year student, the Head should forward this form for approval by the Faculty EIR committee: submit to Ms Zulpha Geyer (Zulpha.Geyer@uct.ac.za; Chem Eng Building, Ph 021 650 4791). NB: A copy of this signed form must be included with the thesis/dissertation/report when it is submitted for examination

***This form must only be completed once the most recent revision EBE EIR Handbook has been read.***

---

Name of Principal Researcher/Student: Ra'eesah Mangera  Department: Electrical Engineering

Preferred email address of the applicant: rmangera@csir.co.za

**If a Student:**  Degree: MScEng  Supervisor: Dr Fred Nicolls

If a Research Contract indicate source of funding/sponsorship: Council for Scientific and Industrial Research (CSIR)

Research Project Title: Gesture Recognition with application in Robotic Control

**Overview of ethics issues in your research project:**

| | | |
|---|---|---|
| **Question 1: Is there a possibility that your research could cause harm to a third party (i.e. a person not involved in your project)?** | YES | NO |
| **Question 2: Is your research making use of human subjects as sources of data?** If your answer is YES, please complete Addendum 2. | YES | NO |
| **Question 3: Does your research involve the participation of or provision of services to communities?** If your answer is YES, please complete Addendum 3. | YES | NO |
| **Question 4: If your research is sponsored, is there any potential for conflicts of interest?** If your answer is YES, please complete Addendum 4. | YES | NO |

If you have answered YES to any of the above questions, please append a copy of your research proposal, as well as any interview schedules or questionnaires (Addendum 1) and please complete further addenda as appropriate. Ensure that you refer to the EIR Handbook to assist you in completing the documentation requirements for this form.

**I hereby undertake to carry out my research in such a way that**

- there is no apparent legal objection to the nature or the method of research; and
- the research will not compromise staff or students or the other responsibilities of the University;
- the stated objective will be achieved, and the findings will have a high degree of validity;
- limitations and alternative interpretations will be considered;
- the findings could be subject to peer review and publicly available; and
- I will comply with the conventions of copyright and avoid any practice that would constitute plagiarism.

**Signed by:**

| | Full name and signature | Date |
|---|---|---|
| Principal Researcher/Student: | Ra'eesah Mangera<br>Signed by candidate | 20/03/2014 |

**This application is approved by:**

| | | |
|---|---|---|
| Supervisor (if applicable): | Signed by candidate Fred Nicolls | 20/03/2014 |
| HOD (or delegated nominee):<br>*Final authority for all assessments with NO to all questions and for all undergraduate research.* | | |
| Chair : Faculty EIR Committee<br>For applicants other than undergraduate students who have answered YES to any of the above questions. | Signed by candidate | 25/04/2014 |

**CSIR Research Ethics Committee**
PO Box 395 Pretoria 0001 South Africa
Tel: +27 12 841 4060
Fax: +27 12 841 2476
Email:R&DEthics@csir.co.za

30 June 2014

Dear: Ms Ra'eesha Mangera

Approval of Protocol: **Testing Gesture Recognition with Application in Robotic Control**

This is to confirm that your Protocol reviewed by the CSIR REC has been approved. The reference number of this research project is REF: 101/2014.

This approval is granted under the condition that:

1. The researcher remains within the procedures and protocols indicated in the proposal, as well as the additions made to the procedures and protocols as indicated in the responses submitted to the questions of the REC, particularly in terms of any undertakings made and guarantees given.
2. The researcher notes that the research must be submitted again for ethical clearance if there is substantial departure from the existing proposal.
3. The researcher remains within the parameters of any applicable national legislation, institutional guidelines and scientific standards relevant to the specific field of research.
4. This approval is valid for one calendar year from the date of this letter.
5. The researcher submit bi-annual progress reports to the REC
6. The researcher immediately alert the REC of any adverse events that have occurred during the course of the study, as well as the actions that were taken to immediately respond to these events.
7. The researcher alert the REC of any new or unexpected ethical issues that emerged during the course of the study, and how these ethical issues were addressed. If unsure how to respond to these unexpected or new ethical issues as they emerge, the researcher should immediately consult with the REC for advice.
8. The researcher submit a short report to the REC on completion of the research in which it is indicated (i) that the research has been completed; (ii) if any new or unexpected ethical issues emerged during the course of the study; and if so, (iii) how these ethical issues were addressed.

We wish you all of the best with your research project.

Kind regards

Dr Mongezi Mdhluli

Signed by candidate

(CSIR REC Chair)

Dr Sandile Ncanana

Signed by candidate

(CSIR REC Secretariat)

# C. Calculating the Intersection of a Circle and Line

Consider a circle centred at point **C** with radius $r$, and a line, **LE**, from point **L** to point **E** (as depicted in Figure C.1).

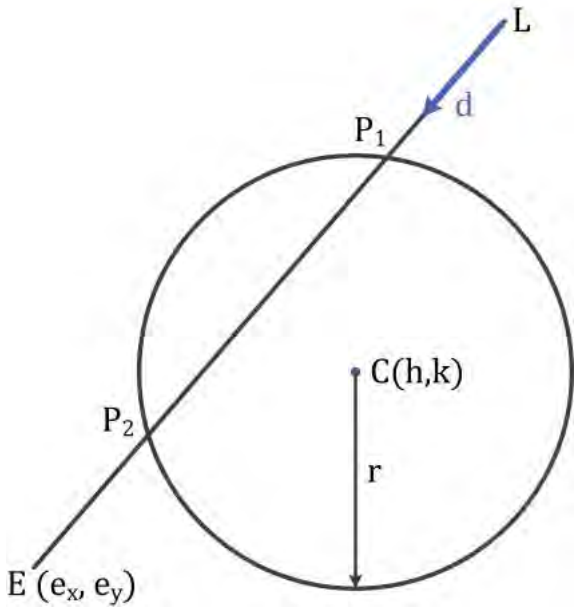



**Figure C.1.** Intersection of line and circle.

The points of intersection $\mathbf{P}_1$ and $\mathbf{P}_2$ are found as follows:
Define the direction vector of **LE**, from start to end, as

$$\mathbf{d} = \mathbf{L} - \mathbf{E}, \tag{C.1}$$

and the vector from the centre of the circle to the start of the line **LE** as

$$\mathbf{f} = \mathbf{E} - \mathbf{C}. \tag{C.2}$$

The parametric equation of the line **LE** is given by:

$$\mathbf{P} = \mathbf{E} + t\mathbf{d}. \tag{C.3}$$

In Cartesian coordinates this is

$$p_x = e_x + td_x, \tag{C.4}$$

$$p_y = e_y + td_y. \tag{C.5}$$

The equation of the circle is given by

$$(x-h)^2 + (y-k)^2 = r^2, \tag{C.6}$$

where $C = (h, k)$ is the centre of the circle.

To find the intersection, substitute Eq. (C.4) and Eq. (C.5) into Eq. (C.6) to obtain

$$(e_x + td_x)^2 - 2(e_x + td_x)h + h^2 + (e_y + td_y)^2 - 2(e_y + td_y)k + k^2 - r^2 = 0. \tag{C.7}$$

Expanding and grouping Eq. (C.7) to obtain

$$t^2(\mathbf{d} \cdot \mathbf{d}) + 2t(\mathbf{d} \cdot (\mathbf{E} - \mathbf{C})) + (\mathbf{E} - \mathbf{C}) \cdot (\mathbf{E} - \mathbf{C}) - r^2 = 0. \tag{C.8}$$

Finally, substituting Eq. (C.2) into Eq. (C.8) to get

$$t^2(\mathbf{d} \cdot \mathbf{d}) + 2t(\mathbf{d} \cdot \mathbf{f}) + \mathbf{f} \cdot \mathbf{f} - r^2 = 0. \tag{C.9}$$

This equation can be solved using the quadratic formula

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}, \tag{C.10}$$

where

$$\begin{aligned} a &= \mathbf{d} \cdot \mathbf{d}, \\ b &= 2(\mathbf{d} \cdot \mathbf{f}), \\ c &= \mathbf{f} \cdot \mathbf{f} - r^2. \end{aligned} \tag{C.11}$$

The variable $t$ can then be substituted into Eq. (C.4) and Eq. (C.5) to obtain the points where the circle and the line intersect.

# D. Live Testing Videos

This chapter includes a description and links to the live testing videos that show the real-time testing of the gesture recognition system.

## D.1. Hand Gesture Recognition

Live testing of the hand gesture recognition system was done using a Rock, Paper, Scissors game. In the game either an image of a rock, paper or scissor is shown. The user is then instructed to perform the gesture that will win, lose or tie against the displayed gesture. Once the correct gesture is displayed a new gesture and instruction is shown on the screen. The objective of the game is to get the maximum number of gestures correct in one minute.

The real-time demo of the game can be found at:

`https://www.dropbox.com/s/h86pikn1o9t3p1r/HandGestureRecognitionRealTime.wmv?dl=0`.

A slowed down version of the demo, where the misclassifications are highlighted may be found at:

`https://www.dropbox.com/s/to9cn5o6j2hjxbu/HandGestureRecognitionSlow.wmv?dl=0`.

## D.2. Upper Body Gesture Recognition

Live testing of the upper body gesture recognition system was done using an Asus Xtion Pro Live Camera and a simulated robot. The video in the link below shows the robot reactions to the dynamic gestures and the far-mode interface.

`https://www.dropbox.com/s/9gmw0eb6dxtw52y/UpperBodyGestureRecognition.wmv?dl=0`